The Astrophysical Journal Supplement Series, 281:35 (27pp), 2025 December https://doi.org/10.3847/1538-4365/ade851

Gamma-Ray Burst Light-curve Reconstruction: A Comparative Machine and Deep Learning Analysis

A. Manchanda[1,2,23], A. Kaushal[3,23], M. G. Dainotti[4,5,6,7,23,24], K. Gupta[8], A. Deepu[9], S. Naqi[10], J. Felix[11], N. Indoriya[12], S. P. Magesh[13], H. Gupta[14], A. Madhan[15], D. H. Hartmann[16], A. Pollo[17,18], M. Bogdan[19], J. X. Prochaska[20], N. Fraija[21], and D. Debnath[22]

[1] Centre for Astrophysics and Supercomputing, Swinburne University of Technology, Victoria 3122, Australia
[2] Department of Physics, Indian Institute of Technology, Hyderabad, Telangana, 502284, India
[3] Department of Computer Science and Engineering, UIET-H, Panjab University, Punjab, 146001, India
[4] National Astronomical Observatory of Japan, 2 Chome-21-1 Osawa, Mitaka, Tokyo 181-8588, Japan
[5] Department of Astronomical Sciences, The Graduate University for Advanced Studies, SOKENDAI, Shonankokusaimura, Hayama, Miura District, Kanagawa 240-0115, Japan
[6] Nevada Center for Astrophysics, University of Nevada, 4505 Maryland Parkway, Las Vegas, NV 89154, USA
[7] Space Science Institute, 4765 Walnut Street, Suite B, Boulder, CO 80301, USA
[8] Department of Computer Science, PSIT College, Kanpur, Uttar Pradesh, 209305, India
[9] Department of Ocean Engineering and Naval Architecture, Indian Institute of Technology Kharagpur, West Bengal, 721302, India
[10] KNIT Sultanpur, Uttar Pradesh, 228118, India
[11] Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India
[12] Department of Electrical Engineering and Computer Science, IISER Bhopal, Madhya Pradesh, 462066, India
[13] Indian Institute of Technology, Madras, 600036, India
[14] MIT WPU School of Computer Science & Engineering, Pune, 411038, India
[15] Visveswaraya Technological University, Belagavi, Karnataka, 590018, India
[16] Department of Physics and Astronomy, Clemson University, Clemson, SC 29634, USA
[17] Astronomical Observatory of the Jagiellonian University in Kraków, Orla 171, 30-244 Kraków, Poland
[18] National Centre for Nuclear Research, 02-093 Warsaw, Poland
[19] Department of Mathematics, University of Wroclaw, Wroclaw, 50-384, Poland
[20] University of California, Santa Cruz, 1156 High Street, Santa Cruz, CA 95064, USA
[21] National Autonoma University of Mexico, Circuito Interior, 04510 Mexico City, Mexico City, Mexico
[22] NIT Agartala, Tripura, 799046, India
Received 2025 March 25; revised 2025 May 31; accepted 2025 June 2; published 2025 November 14

Abstract

Gamma-ray bursts (GRBs), observed at high-z, are probes of the evolution of the Universe and can be used as cosmological tools. Thus, we need correlations with small dispersion among key parameters. To reduce such a dispersion, we mitigate gaps in light curves (LCs), including the plateau region, key to building the two-dimensional Dainotti relation between the end time of plateau emission (T_a) and its luminosity (L_a). We reconstruct LCs using nine models: multilayer perceptron (MLP), Bi-Mamba, Fourier transform, Gaussian process-random forest hybrid, bidirectional long short-term memory, conditional generative adversarial networks, SARIMAX-based Kalman filter, Kolmogorov–Arnold networks, and Attention U-Net. These methods are compared to the Willingale model (W07) over a sample of 521 GRBs. MLP and Attention U-Net outperform other methods, with MLP reducing the plateau parameter uncertainties by 37.2% for $\log T_a$, 38.0% for $\log F_a$, and 41.2% for α (the post-plateau slope in the W07 model), achieving the lowest fivefold cross-validation mean squared error (MSE) of 0.0275. Attention U-Net achieved the lowest uncertainty of parameters, a 37.9% reduction in $\log T_a$, a 38.5% reduction in $\log F_a$, and a 41.4% reduction in α, but with a higher MSE of 0.134. Although Attention U-Net achieves the largest uncertainty reduction, the MLP attains the lowest test MSE while maintaining comparable uncertainty performance, making it the more reliable model. The other methods yield MSE values ranging from 0.0339 to 0.174. These improvements in parameter precision are needed to use GRBs as standard candles, investigate theoretical models, and predict GRB redshifts through machine learning.

Unified Astronomy Thesaurus concepts: Gamma-ray bursts (629)

Materials only available in the online version of record: machine-readable table

1. Introduction

Gamma-ray bursts (GRBs) are brief and highly luminous astrophysical phenomena detectable at remarkable distances (for a review, see P. Kumar & B. Zhang 2015), with observations extending up to redshift $z = 9.4$ (A. Cucchiara et al. 2011). This property makes GRBs exceptional cosmological tools for probing the Universe's early evolution. A thorough analysis of GRBs also yields essential information about Population III stars, the first generation formed during the epoch of reionization. GRB emission is usually observed in two episodes: the prompt and the afterglow. The prompt phase is interpreted by internal shell collisions or magnetic reconnection and is characterized by a short duration and high energy. Hence, this is detected predominantly in γ-rays and

[23] These authors contributed equally to this work.
[24] This author is the corresponding author.

X-rays and occasionally in optical wavelengths (C. H. Blake et al. 2005; W. T. Vestrand et al. 2005; G. Beskin et al. 2010; E. S. Gorbovskoy et al. 2012; W. T. Vestrand et al. 2014). During the afterglow phase, which follows the primary episode, the relativistic jet impacts the circumstellar environment, transferring part of its energy. The afterglow is characterized by a long duration and energies that span a broad spectrum, including X-ray, optical, and sometimes radio bands (E. Costa et al. 1997; J. van Paradijs et al. 1997; L. Piro et al. 1998; N. Gehrels et al. 2009).

The Neil Gehrels Swift Observatory (Swift; N. Gehrels et al. 2004) is vital for detecting the temporal properties of GRBs. The Swift Burst Alert Telescope, which operates in the range of 15–150 keV (S. D. Barthelmy et al. 2005), plays a crucial role in quickly detecting prompt emission. It enables a rapid follow-up of the afterglow through the X-ray Telescope instrument, which covers the 0.3–10 keV range (D. N. Burrows et al. 2005), and the Ultra-Violet Optical Telescope (170–600 nm (P. W. A. Roming et al. 2005). Moreover, Swift's rapid multiwavelength afterglow follow-up has revealed new characteristics in GRB LCs (G. Tagliaferri et al. 2005; J. A. Nousek et al. 2006; E. Troja et al. 2007).

Most X-ray LCs exhibit rapid decay in flux after the prompt episode, occasionally followed by flares and/or a plateau (P. T. O'Brien et al. 2006; J. A. Nousek et al. 2006; B. Zhang et al. 2006; E.-W. Liang et al. 2007; T. Sakamoto et al. 2007; R. Willingale et al. 2007; M. G. Dainotti et al. 2008, 2010, 2016, 2017; H. Dereli-Bégué et al. 2025). The plateau observed in GRB LCs can be modeled using a broken power law (BPL; B. Zhang et al. 2006, 2007; J. L. Racusin et al. 2009), a smooth BPL, or the phenomenological model proposed by R. Willingale et al. (2007, hereafter W07). The W07 model determines critical parameters such as the time at the end of the plateau, T_a, the corresponding flux, F_a, and the temporal index after the plateau, α_a. On the other hand, the BPL model provides T_a, F_a, and the slope of the LC during the plateau, α_1, and after the plateau, α_2. Section 2.1 details the W07 model and its parameters.

The plateau phase is frequently interpreted using the magnetar model (B. Zhang & P. Mészáros 2001; A. Rowlinson et al. 2014; N. Rea et al. 2015; G. Stratta et al. 2018), which attributes the emission to dipole radiations generated by the rotational energy of a newly formed neutron star (NS). According to this model, the plateau ends when the NS reaches its critical spin-down timescale. Uncertainties in determining T_a are often related to the magnetar spin period and uncertainties of the magnetic field. Therefore, precise measurements of T_a are essential to verify the validity of this model. The plateau phase, exhibiting more consistent features across various GRBs, such as length and flatness, has attracted attention because of its potential to establish relevant correlations with the plateau parameters and their application as cosmological tools. E.-W. Liang et al. (2007); M. G. Dainotti et al. (2008, 2010, 2011, 2013, 2015, 2017); L. Li et al. (2018); C.-H. Tang et al. (2019); L. Zhao et al. (2019); and F. Y. Wang et al. (2022) have explored the luminosity at the end of the plateau, $L_{X,a}$ versus its rest-frame time $T^*_{X,a}$ (known as the Dainotti relation or the 2D L-T relation).[25] The 2D relation has also been identified in the optical plateau emissions (M. G. Dainotti et al. 2020c, 2022d).

[25] The rest-frame time is denoted by an asterisk.

Within the theoretical magnetar framework, A. Rowlinson et al. (2014) showed that the X-ray Dainotti relation is reproduced with a slope for $L_{a,X}$-$T^*_{a,X}$ of -1. This correlation has been applied in cosmological research, such as the development of the GRB Hubble diagram, which extends to redshifts greater than $z > 8$ (V. F. Cardone et al. 2009, 2010; M. G. Dainotti et al. 2013; S. Postnikov et al. 2014).

The 2D L-T relation has been further expanded by incorporating the peak prompt luminosity, $L_{X,\mathrm{peak}}$, resulting in the Dainotti 3D relation (M. G. Dainotti et al. 2016, 2017, 2020b, 2022d). This 3D relation has also been successfully applied to constrain cosmological parameters (S. Cao et al. 2022a, 2022b; M. G. Dainotti et al. 2022c, 2022b, 2023a). Importantly, M. G. Dainotti et al. (2022b) demonstrated that reducing the uncertainties associated with the plateau emission parameters by 47.5% could achieve the same precision for the cosmological value of Ω_M quoted in M. Betoule et al. (2014) in just 8 years as we have calculated in M. G. Dainotti et al. (2022a) and A. Narendra et al. (2025), if we consider the optical sample and the addition of GRBs for which the redshift was inferred (M. Dainotti et al. 2024a, 2024b). This improvement could be realized immediately, compared to the precision requiring 22 more years of observations under current rates and parameter uncertainties (M. G. Dainotti et al. 2022b), which highlights the significant potential of a more reliable light curve reconstruction (LCR) approach, as it could substantially accelerate progress to reach the same precision achieved by Type 1a supernovae (SNe 1a; for details, see M. G. Dainotti et al. 2020b).

In addition, GRB LCs with temporal gaps present significant challenges in testing theoretical models, such as the standard fireball model (T. Piran 1999; A. Panaitescu & P. Kumar 2000). This model is typically evaluated using closure relations (W07; P. Evans et al. 2009; J. L. Racusin et al. 2009; P. Kumar & R. B. Duran 2010; D. Tak et al. 2019; G. Ryan et al. 2020; G. P. Srinivasaragavan et al. 2020; M. G. Dainotti et al. 2021), which involve the temporal (α) and spectral (β) index of the afterglow. The value of α can correspond to α_1 or α_2, depending on the segment of the LC analyzed, while β is measured for the same time window as α. An accurate assessment of these factors is essential to evaluate the fireball model and categorize GRBs according to their morphology.

However, gaps due to satellite orbital periods, lack of fast follow-up studies, and factors such as meteorological turbulence and instrumental errors complicate these measurements. To address these challenges, further improvement in understanding GRBs requires extensive data coverage and the development of a reliable taxonomy of GRB classes. Such efforts could considerably improve studies on GRB populations, their cosmological evolution, emission mechanisms, and/or progenitors. Lastly, another valuable application involves the utilization of reconstructed LCs to train machine learning models to estimate the redshifts of GRBs (G. Bargiacchi et al. 2025).

Thus, here we tackle the LCR, which provides a novel approach to address the challenge of temporal gaps in LCs, which is a continuation of a previous effort by some of us (M. G. Dainotti et al. 2023b). That paper introduced a stochastic reconstruction technique using existing models and Gaussian processes (GPs), reducing uncertainties for key GRB parameters.

Recent advances in machine learning have further pushed the boundaries of LCR. For example, M. Demianenko et al. (2023) examined the application of neural network-based methods, including Bayesian neural networks, multilayer perceptrons, and normalization flows, to obtain approximate findings of a single LC. These methods were tested on the LCs of SNe 1a taken from the Zwicky Transient Facility Bright Transient Survey and simulated PLAsTiCC. These methods demonstrated that even with limited observations, neural networks could significantly enhance the quality of the reconstruction in comparison to state-of-the-art models. Moreover, they are computationally efficient, outperforming GPs in terms of speed, and were found to be effective for subsequent tasks such as peak identification and transient classification.

Building on these advancements, this work further utilizes deep learning techniques to improve the GRB LCR. In particular, we utilized Bi-Mamba, multilayer perceptron (MLP), Fourier transform, ensemble GPs with random forest (RF) models, bidirectional long short-term memory (Bi-LSTM), conditional generative adversarial networks (CGAN), SARIMAX-based Kalman filters, Kolmogorov–Arnold networks (KANs), and the hybrid model of U-Net with attention mechanism (Attention U-Net). This study highlights the strengths and limitations of these models and performs a comparative analysis to assess their performance.

This paper is divided into the following sections: Section 2 provides details on the data set and the various models employed to reconstruct GRB LCs. The uncertainty, performance, and outliers are discussed in Section 4. Section 5 provides a synopsis and conclusions on the observed efficacy of each model.

2. Methodology

2.1. Data Set and the Willingale Model

This research analyzed a data set comprising 521 GRBs. The initial sample comprises 455 GRBs detected by Swift between 2005 January and 2019 August, as compiled by G. P. Srinivasaragavan et al. (2020) and M. G. Dainotti et al. (2020b). This data set is then enlarged by an additional 14 GRBs observed by Fermi-LAT, as presented in M. Dainotti et al. (2025), and a further 78 GRBs detected by Swift between 2019 and 2023, as reported by A. Narendra et al. (2025). After accounting for two overlapping GRBs between the original and extended samples, the final data set consists of 521 unique GRBs. The data set includes 230 GRBs with known redshifts and 315 without redshift information, as described in M. Dainotti et al. (2024a, 2024b), A. Narendra et al. (2025). All GRBs in the sample, including flares (about 31.5% of our sample), exhibit temporal gaps in their LCs arising from various factors. To address irregular gaps in GRB LCs, we implemented a gap-aware reconstruction in log-time space. A minimum gap threshold of 0.05 was imposed. The number of reconstructed points was adaptively set as a fraction of the original LC size: 5% for LCs with more than 500 points, 10% for those with 250–500 points, 30% for 100–250 points, and 40% for fewer than 100 points (minimum of 20 points). Extremely sparse LCs (about 5% of the sample with <26 points) with very large gaps, reconstruction was not feasible. Thus, the data set was reduced to 521 GRBs.

The W07 function models the LCs of GRBs, defined in Equation (1) and introduced by W07:

$$f(t) = \begin{cases} F_i \exp\left(\alpha_i\left(1 - \frac{t}{T_i}\right)\right)\exp\left(-\frac{t_i}{t}\right) \text{ for } t < T_i, \\ F_i\left(\frac{t}{T_i}\right)^{-\alpha_i}\exp\left(-\frac{t_i}{t}\right) \text{ for } t \geqslant T_i. \end{cases} \quad (1)$$

Here, the symbols T_i and F_i refer to time and flux, respectively, in our case, at the end of the plateau emission, α_i is the temporal index parameter after T_i. The parameter t_i corresponds to the onset of the rise phase, and the afterglow emission is represented by t_a. We have shown in previous papers that the onset is usually compatible with 0 (W07).

The LCs analyzed in this study have had the prompt emission segment removed because of its significant variability and the challenges in modeling it effectively. Consequently, the data set is limited to the plateau and afterglow phases.

The GRBs were subsequently categorized into four distinct classes based on the features of their afterglow emission, using the classification methodology established in M. G. Dainotti et al. (2023b):

1. *Good GRBs.* The afterglow is a good approximation with the W07 model, representing 55% of the data set.
2. *Flares/bumps.* GRBs in this category show flares and bumps throughout the afterglow phase, accounting for 24% of the total.
3. *Break.* GRBs with a single break observed toward the end of the LC make up 13% of the sample.
4. *Flares/bumps + double break.* GRBs that display a combination of flares/bumps and a double break constitute 7.5% of the data set.

The first four panels of Figure 1 illustrate the LCs for the four identified classes and their corresponding W07 model fits. Our analysis relies on the analysis of GRB afterglow LCs by comparing each LC with the W07 fitted model. The parameters for each GRB LC have been derived from the work of G. P. Srinivasaragavan et al. (2020). The blue points represent the trimmed data, where the prompt emission phase has been removed.

2.2. Reconstruction with the Willingale Model

In this section, we adhere to the methodology described in M. G. Dainotti et al. (2023b) to reconstruct the LCs using the W07 model. However, our approach further extends the analysis by including all classes of GRBs, allowing for a more extensive study of the model across different categories of GRBs.

The flux residual for each LC is calculated as the difference between the logarithmic flux of the original LC and the logarithmic flux value predicted by the W07 fit at a given time t (in log10 scale):

$$\log_{10} F_{\rm res} = \log_{10} F_{\rm obs}(t) - \log_{10} f(t) \quad (2)$$

where $\log_{10} F_{\rm res}$ is the flux computed as the residual, while $F_{\rm obs}$ is the observed flux, and $f(t)$ is the functional form of W07 at the given time t.

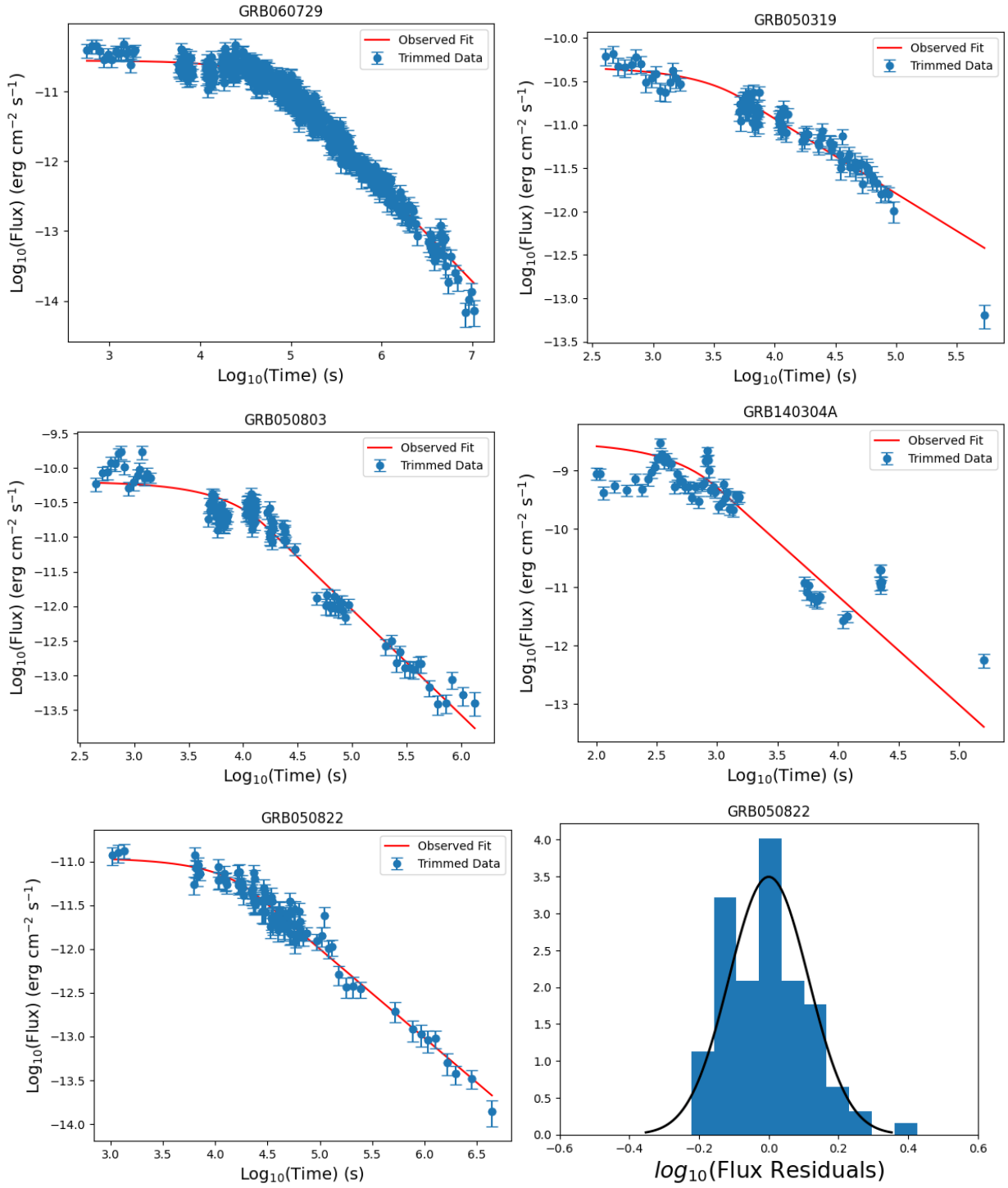


Figure 1. Rows 1 and 2 describe the GRB LCs divided into four classes depending on the afterglow feature: (i) good GRBs (row 1, left); (ii) GRB LCs with a break toward the end (row 1, right); (iii) flares or bumps in the afterglow (row 2, left; and (iv) flares or bumps with a double break toward the end of the LC (row 3, right). Row 3's left plot shows the LC of GRB 050822, starting from the plateau emission, and the best-fit W07 model is shown in red. The right plot of row 3 shows the log (flux) residual histogram, and the best-fit Gaussian distribution is displayed in black.

We then perform the same steps described, generating histograms of the residuals for each GRB and fitting a Gaussian distribution, as a first approximation, to those residuals as presented in the bottom right panel of Figure 1. In our analysis, we followed the same methodology as outlined in M. G. Dainotti et al. (2023b), where the fitting was performed using the Gaussian, since in most of the cases, the Gaussian distribution is quoted to be the best fit in the package *FindDistribution*. However, there are several cases in which other distributions may have fitted better, but to provide a uniform treatment for simplicity and to better compare our analysis with that of M. G. Dainotti et al. (2023b), we have adopted the same methodology. In future work, we will use in each case the best-fit distribution for each GRB. The reconstructed flux at each time point is calculated by adding noise, sampled from the residual distribution, to the flux

predicted by the W07 model:

$$\log_{10} F_{\rm recon}(t) = \log_{10} f(t) + (1 + n) \times \text{RVN}. \quad (3)$$

where $\log_{10} F_{\rm recon}(t)$ represents the reconstructed flux at time t, $f(t)$ denotes the W07 flux at time t, n is the noise level, and RVN refers to the random variate sampled from the normal (Gaussian) distribution.

We evaluated the reconstruction at 10% and 20% noise levels. This method ensures that the reconstructed data points are statistically consistent with the best-fit W07 model as the initial observed points. We reconstruct time series using a logarithmic distribution via the *geomspace* function and perform 100 Markov Chain Monte Carlo (MCMC) calculations for each GRB to guarantee the stability of the model. The combined data (original and reconstructed points) is then refitted using a least-squares regression with the *minimize()* function from the *lmfit* library, providing the new fitting parameters and uncertainties of the reconstructed LCs.

2.3. The Machine Learning Approach

Machine learning models are algorithms that learn from the data, recognize patterns, and give predictions on unseen data. One of the most widely used machine learning techniques is the artificial neural network (ANN; J. Heaton 2018; S.-C. Wang 2003). These are inspired by how the human brain (neurons) processes information. Just as the brain consists of interconnected neurons that help us think and learn, a neural network consists of layers of artificial neurons that process and learn from data. A neural network consists of the input, hidden, and output layers. The input layer is the first layer that receives the raw input data. In this layer, no computations are performed, as it is responsible for passing the data to the next layer. The output layer is the final layer that gives the output predictions of the hidden layer. The hidden layers consist of artificial neurons to analyze the data. Each connection between neurons has a weight, determining the input value's importance. A bias is an extra value added to fine-tune and adjust the predictions. At first, the weights and biases are initialized randomly, but they are adjusted to improve the prediction as the model trains. The model undergoes training over a series of epochs, progressively adjusting its weights and biases within each epoch to minimize the errors it commits.

Each neuron gives output as a weighted sum of the inputs (containing biases). However, if we only use this method, the network can learn only linear relationships, which limits its ability to capture complex patterns. To overcome this, we apply activation functions, which introduce nonlinearity and allow the network to learn more advanced relationships in data. After applying the activation function, the neuron passes the result to the next layer, allowing the network to process data step by step until it reaches the output layer. The performance of neural networks greatly depends on hyperparameters, which are the configuration variables that must be fixed before training. These may include the number of hidden layers, neurons, epochs, etc. The technical terms are summarized in Table 1.

This study uses various machine learning models to reconstruct the GRB LCs. The MLP is the most basic type of neural network. It is a specific type of ANN with fully connected layers, meaning that each neuron in one layer is connected to every neuron in the next layer. We have compared this with other advanced neural networks that extend the basic neural network differently. For instance, Bi-LSTM is designed to work with sequences. A standard neural network treats every input independently, but Bi-LSTM "remembers" the past information to process the new ones. Sometimes, not all parts of the data are equally important. We introduce Attention U-Net, where the "attention" mechanisms help the network focus on the most relevant parts while ignoring less important details. By incorporating probability into the model parameters (weights and biases), we use GPs to predict output and uncertainty. In most neural networks, how neurons process information is fixed, meaning that the transformation applied to the data remains the same throughout training. KAN improves this by adapting the activation functions during learning. Other methods, such as SARIMAX and Fourier, use statistical techniques to analyze time-series patterns.

To ensure robustness and consistency, the proposed models share several common steps in both preprocessing and result calculation, as outlined below:

1. *Individual GRB handling.* Due to the distinct nature of each GRB's afterglow LC, each GRB was trained and reconstructed individually rather than using a generalized model. This approach preserves each burst's unique temporal and flux evolution patterns. The Appendix explains this in more detail. For every model, the following process was applied: time values (inputs) and corresponding flux values (labels) for each GRB were $\log_{10}$ transformed and used for training.
2. *Input and output.* The model is trained using time values whose lengths vary depending on each GRB paired with its corresponding target flux values. During reconstruction, the model is given a missing time window (gap length greater than 0.03 in log timescale units) to predict the intermediate flux values.
3. *Parameter tuning.* We selected a subset of 16 GRBs for hyperparameter tuning, with four GRBs randomly chosen from each of the four GRB classes. For each of these 16 GRBs, we held out 20% of the data as a validation set and used the remaining 80% for training. Hyperparameter optimization was conducted individually for each GRB using Bayesian optimization via the *Optuna* module (T. Akiba et al. 2019), with the goal of minimizing the validation loss on the 20% hold-out set. Importantly, we did not use early stopping in this process or any other model training stage. Instead, the number of training epochs was itself treated as a tunable hyperparameter and optimized during the *Optuna*-based tuning phase. For each GRB, the optimal number of epochs (along with other hyperparameters) was selected based on validation loss. Once optimal hyperparameters were obtained for each of the 16 GRBs, we computed their average across the 16 runs. For parameters requiring integer values, we rounded the averaged result to the nearest integer.

   This resulted in a fixed set of hyperparameters (including the number of epochs) for each model architecture. These were then used to train each model on the complete data set of 521 GRBs. This tuning process was performed separately for each model type evaluated in the study.

   The following hyperparameters were tuned depending on the model architecture:

Table 1
Definition of Machine Learning Terms

Machine Learning Terms	Definition
Neural network	The machine learning model inspired by our brain consists of layers of interconnected "neurons" to process information and learn patterns from the data.
Hyperparameters	Variables that need to be optimized to control the training process. In our study, we have used *GridSearchCV* and *Optuna* to optimize.
Activation function	It introduces nonlinearity to the outputs of the neurons to learn complex data.
Rectified linear unit (ReLU)	An activation function that outputs the input directly if it is positive, and zero otherwise, mathematically represented as max(0,x).
LeakyReLU	A variant of ReLU introduces small, nonzero slope values for negative inputs.
Sigmoid activation function	A mathematical function that maps input values to a range between 0 and 1.
Loss function	It quantifies the difference between the model's prediction and the actual values.
Mean squared error (MSE)	The loss function that measures the average squared difference between predicted and true values.
Optimizer	An algorithm that adjusts the model parameters (like weights and biases) to minimize the loss function. We use the Adam optimizer.
Backpropagation	Technique to adjust weights and biases to minimize the loss function.
Learning rate	A hyperparameter that controls the rate at which the model updates its parameters.
Epoch	One full pass through the entire data set during training, allowing the model to learn patterns and adjust the parameters
EarlyStopping	A technique to prevent overfitting by stopping training when the validation loss stops improving after a set number of iterations.
Patience	A parameter used in early stopping that defines how many epochs to wait after the last improvement before stopping training.
He uniform initializer	A weight initialization method that draws weights from a uniform distribution, scaled based on the number of input neurons.
Cross validation (CV)	Evaluates the model's performance by dividing the data set into multiple subsets for training and testing.
Outliers	The GRBs that display uncertainty greater than 100% for a particular model when fitted with the W07 fitting function.

a. *MLP.* Number of hidden layers, neurons per layer, number of epochs.
b. *Attention U-Net.* Number of filters per layer, kernel size, number of convolutional blocks, learning rate, batch size, and number of epochs.
c. *Bi-LSTM.* Number of Bi-LSTM layers, number of hidden units, activation function, number of epochs, and learning rate.
d. *Bi-MAMBA.* d_state, dropout rate, d_expand, number of epochs, learning rate.
e. *GP-RF hybrid.* Number of estimators, maximum tree depth, minimum samples for node splitting, and minimum samples per leaf.
f. *CGAN.* Generator and discriminator hidden dimensions, number of epochs, and learning rate.
g. *KAN.* Width, grid, number of epochs, and learning rate.

4. *Min-max scaling.* For each GRB, the data is scaled to the [0, 1] range using standard min-max normalization (4). However, scaling is not applied to Attention U-Net and MLP models, as performing some tuning in several GRBs showed a decrease in performance. Additionally, Attention U-Net inherently includes multiple batch normalization layers, while the MLP, being a relatively simpler network, is less susceptible to the weights of the model either increasing exponentially or going toward zero, namely, exploding or vanishing gradients.

$$X_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}};\ i \in D, \tag{4}$$

where $X_{\min}$ and $X_{\max}$ are the minimum and the maximum value in D.

5. *Uncertainty estimation and confidence interval.* The uncertainty in the reconstructed flux values was estimated by fitting a Gaussian distribution to the flux residuals, defined as the difference between the original data and the reconstructed data. This Gaussian distribution provides a probabilistic model of the noise. Random noise samples were drawn from this distribution and added to the model's mean prediction to generate realistic flux variations. To account for variability, we performed Monte Carlo simulations with 1000 iterations. For each iteration, random noise was sampled for each reconstructed data point and added to the corresponding flux value. This process resulted in multiple realizations of the reconstructed LC, capturing the statistical uncertainty of the predictions. The mean of these realizations was computed to represent the final reconstructed flux. The 95% confidence intervals were determined by calculating the 2.5th and 97.5th percentiles, denoted with P, of the realizations for each time point, providing a reliable measure of uncertainty for the reconstructed data:

$$\mathrm{CI}_{95} = [\mathrm{percentile}_{2.5},\ \mathrm{percentile}_{97.5}]. \tag{5}$$

6. *Fivefold CV.* After fixing the hyperparameters for each model, fivefold CV is used as an alternate metric to evaluate the performance of each model. Rather than splitting the entire data set of 521 GRBs into separate training and testing sets, we apply CV to each GRB

separately. For each GRB, its time-flux data points are first randomly shuffled and then partitioned into five approximately equal-sized, nonoverlapping subsets (folds). In each of the five iterations, four folds (80% of the data) are used to train the model, and the remaining fold (20%) is used to validate it (R. Kohavi 1995). This ensures that all data points are used for both training and validation across the five folds. After training, the model predicts flux values at the time points in the validation fold. These predicted fluxes are inversely transformed back to the original scale before calculating the MSE, as it depends on the scale. The MSE loss is averaged across the folds and then across the 521 GRBs. CV was used solely to evaluate model performance, not for tuning model parameters or minimizing training error.

7. *Final model.* After evaluating model performance using fivefold CV on each GRB independently, we proceed to train a final model for each GRB. Since the hyperparameters are fixed for every model, we train the final model using the entire available LC information (time-flux values) of each GRB, without any train-test split. This model is then used to reconstruct missing regions (gaps greater than 0.03 in log timescale) in the LC of that GRB. Then, we use this result to calculate the reduction in parameter uncertainty.

2.3.1. GP

GP regression (C. E. Rasmussen 2003) is a probabilistic, nonparametric approach designed for regression tasks. The probabilistic nature of the GP model allows for the likelihood of the prediction instead of producing a singular prediction. The method relies on Bayesian inference, where the prior based on trends in the observed data is updated to compute a posterior that aligns with the data. The model prediction is bound by the confidence interval, which defines the region of the likelihood of the prediction.

Let X be a set in $\mathbb{R}^d$. A GP is a collection of random functions, $f(x)$, defined over the input space X set in $\mathbb{R}^d$. For any set of points in $x_1,....,x_n \in X$, the values $(f(x_1),\ldots,f(x_n))$ follow a joint multivariate Gaussian distribution.

Therefore, we can characterize the GP by its mean function,

$$m(x) = \mathbb{E}[f(x)], \tag{6}$$

and its covariance function,

$$\boldsymbol{K}(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))]. \tag{7}$$

A key component of GP is the kernel, or covariance function $\boldsymbol{K}(x, x')$, which measures the similarity between data points. In machine learning, the most popular kernel function is Gaussian radial basis function (RBF) kernel, which is given by

$$K_\gamma(x, x') = \exp\left(-\frac{\|x - x'\|_2^2}{\gamma^2}\right), \tag{8}$$

where $\|x - x'\|_2^2$ denotes the squared Euclidean distance between x and x', for $x \in \mathbb{R}^d$ and $x' \in \mathbb{R}^d$, and the parameter γ determines the length scale of the associated hypothesis space of the functions. As γ increases, the kernel and induced functions change less rapidly and thus become "smoother" (while they are always infinitely differentiable).

To capture observational uncertainty, the RBF kernel is combined with the white noise kernel, defined as

$$\boldsymbol{K}(x, x') = \sigma^2\delta_{xx'}, \tag{9}$$

where $\delta_{xx'}$ is the Kronecker delta function (1 if $x = x'$, 0 otherwise), and σ^2 represents the variance of the noise.

For our analysis, we utilize an RBF kernel combined with a white noise kernel because we assume that the flux noises are independent and identically normally distributed. The RBF captures the smooth, continuous relationships inherent in the data, and the white kernel accounts for observational noise, ensuring reliability in the model predictions.

The formulation of the kernel is as follows:

$$\begin{aligned}\text{kernel} = {} & \text{RBF}(s_1 = 10, b_1 = [10^{-2}, 10^3]) \\ & + \text{whitekernel}(s_2 = 1, b_2 = [10^{-5}, 10]).\end{aligned} \tag{10}$$

Here, s_1 represents the length_scale, b_1 the length_scale_bounds, s_2 the noise_level, and b_2 the noise_level_bounds.

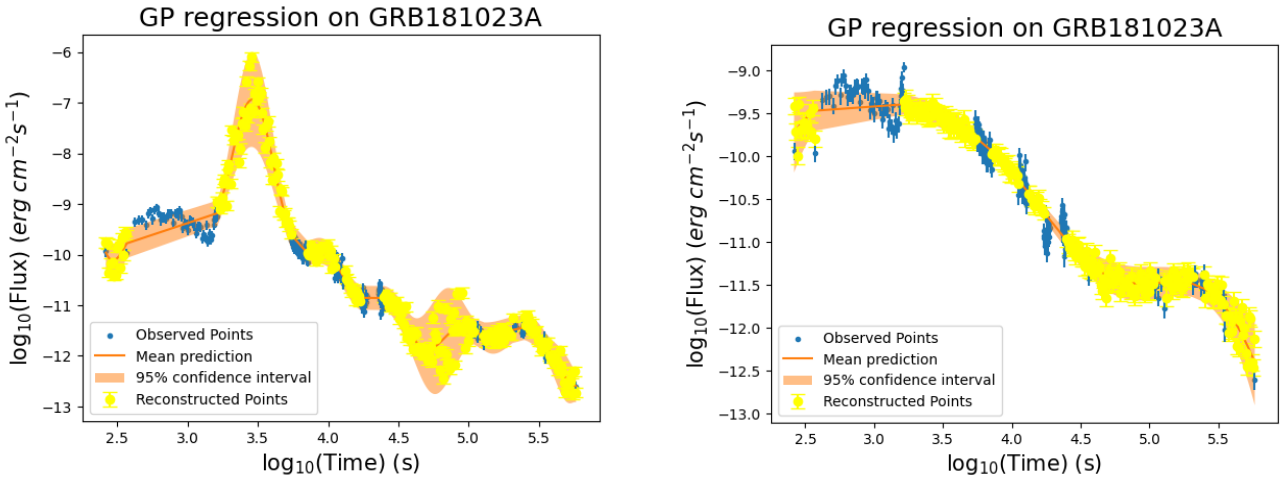
To implement GPs, we utilized the *GaussianProcessRegressor* from `scikit-learn`, enabling prediction with prior knowledge of the GP without its prior fitting. The standardized prior indicates that the mean is centered at zero. The GP regression model was fitted with the built-in *fit* function to our data, and the reconstruction of data points was performed using the built-in *predict* function.

We have selected a 95% confidence interval to reconstruct the LC using GP. Using the form of the function derived by the Gaussian regressor (built-in function in Python), we avoid the distribution of the data points in the GPs by conducting 100 MCMC simulations of the reconstructed LC for each distinct LC. The value and its corresponding uncertainty were then randomly chosen.

In our data, out of the total sample of 521 GRBs, a tiny subset containing nine GRBs with redshifts and 15 GRBs without redshifts exhibited significant temporal gaps, defined as time intervals greater than 0.5 (in $\log_{10}$ scale). As Figure 2(a) shows, the GP regressor overfitted these data points in both gaps, resulting in erratic predictions and extensive 95% confidence intervals. This inconsistent behavior suggested the need to fine-tune two primary hyperparameters: *alpha* and *n_restarts_optimizer*, by employing *GridSearchCV*; *alpha* is a value added to the kernel matrix's diagonal during fitting. This hyperparameter helps avoid potential numerical issues, shown in Figure 2(a), by ensuring the matrix remains positive and definite. Furthermore, *alpha* can be interpreted as the additional Gaussian measurement noise variance on the training observations. The default value is 1e-10. A more considerable *alpha* value is frequently required to avoid overfitting in cases where the data is noisy. *n_restarts_optimizer* denotes the number of times the optimizer restarts to find the kernel parameters that maximize the log-marginal likelihood. This procedure guarantees a more reliable optimization of the kernel's parameters and aids in avoiding local minima.

GridSearchCV uses a fivefold CV to assess model performance as it iteratively searches through a predefined set of hyperparameters. It exhaustively tests all possible combinations of the specified hyperparameters and selects the combination that results in the best performance according to a chosen metric. The *n_restarts_optimizer* is set to 20, and the *alpha* hyperparameter is set to 0.1.

After tuning the hyperparameters, we fit the GP regressor to the data. The optimized model improved the reconstruction of



(a) Without Tuning (b) With Tuning

Figure 2. Comparison of LCs before and after hyperparameter tuning.

the LCs, as shown in Figure 2(b). It might appear that Figure 2(a) offers a better reconstruction than Figure 2(b), as it reconstructs the flare in the first temporal gap. However, this is misleading as very large flares like this have been unusual in the sample of the 521 GRBs. Additionally, the confidence intervals in Figure 2(a) are significantly broader due to the larger temporal gap, reflecting high uncertainty. After hyperparameter optimization, the model shown in Figure 2(b) not only yields smoother and more stable predictions but also exhibits reduced uncertainty, as reflected in narrower confidence intervals.

2.3.2. MLP

The MLP model offers a simple yet effective approach to reconstructing the LCs. By combining the best-fitting curve provided by the model and the noise provided by the Gaussian method, we get an excellent estimate of the flux values and a reduction in uncertainty (as shown in Section 4). This method is helpful since it allows for as much accuracy as the other models while being more straightforward than the other methods.

The neural network was designed to capture the nonlinear relationship between time and flux in the logarithmic space, allowing for smooth reconstruction of LCs. The model has eight layers in total, including the input and output layers. The first layer is the input layer (zeroth layer), which accepts a singular $\log_{10}$(time) value. In every epoch, the entire LC is passed through the model exactly once. The following six layers are the hidden layers (first to sixth layers), and each hidden layer has a varying number of neurons to ensure that the model learns the nonlinear relationship between the flux and time. The hidden layers from the first to the last have 32, 64, 128, 64, 32, and 16 neurons each. Each layer uses the ReLU activation function because it helps enhance the rate at which the network learns the nonlinear relationship between the target and feature variables (A. Agarap 2018). The last layer is the output layer with a single neuron, which makes the final prediction of the $\log_{10}$(flux) value corresponding to the $\log_{10}$(time) value, which was passed into the input layer. The model is represented in Figure 3(a).

Each hidden layer performs the following operation:

$$Z^{(l)} = \boldsymbol{W}^{(l)} \cdot \boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)},$$
$$h^{(l)} = \mathrm{ReLU}(Z^{(l)}), \tag{11}$$

where $W^{(l)}$ and $b^{(l)}$ are the weight and bias matrices of the lth layer. $h^{(l-1)}$ is the activation output of $(l-1)$th layer and $Z^{(l)}$ is the preactivation output of lth layer.

The kernel initialization of each hidden layer uses the *He uniform* (K. He et al. 2015) initializer method. The *He uniform* initializer initializes weights by sampling from a uniform distribution within the range of

$$\text{range} = \left[-\sqrt{\frac{6}{n_{\text{in}}}}, \sqrt{\frac{6}{n_{\text{in}}}} \right], \tag{12}$$

where n_{in} is the number of input units (neurons) to the layer. This ensures that the weights are small enough to prevent exploding gradients while still being large enough to avoid vanishing gradients. It is particularly suited for the ReLU activation function (K. He et al. 2015) since it does not saturate like sigmoid or tanh activation functions. The ReLU activation function, defined as $\max(0, x)$, outputs zero for negative values and keeps positive values unchanged, making it efficient for training deep networks.

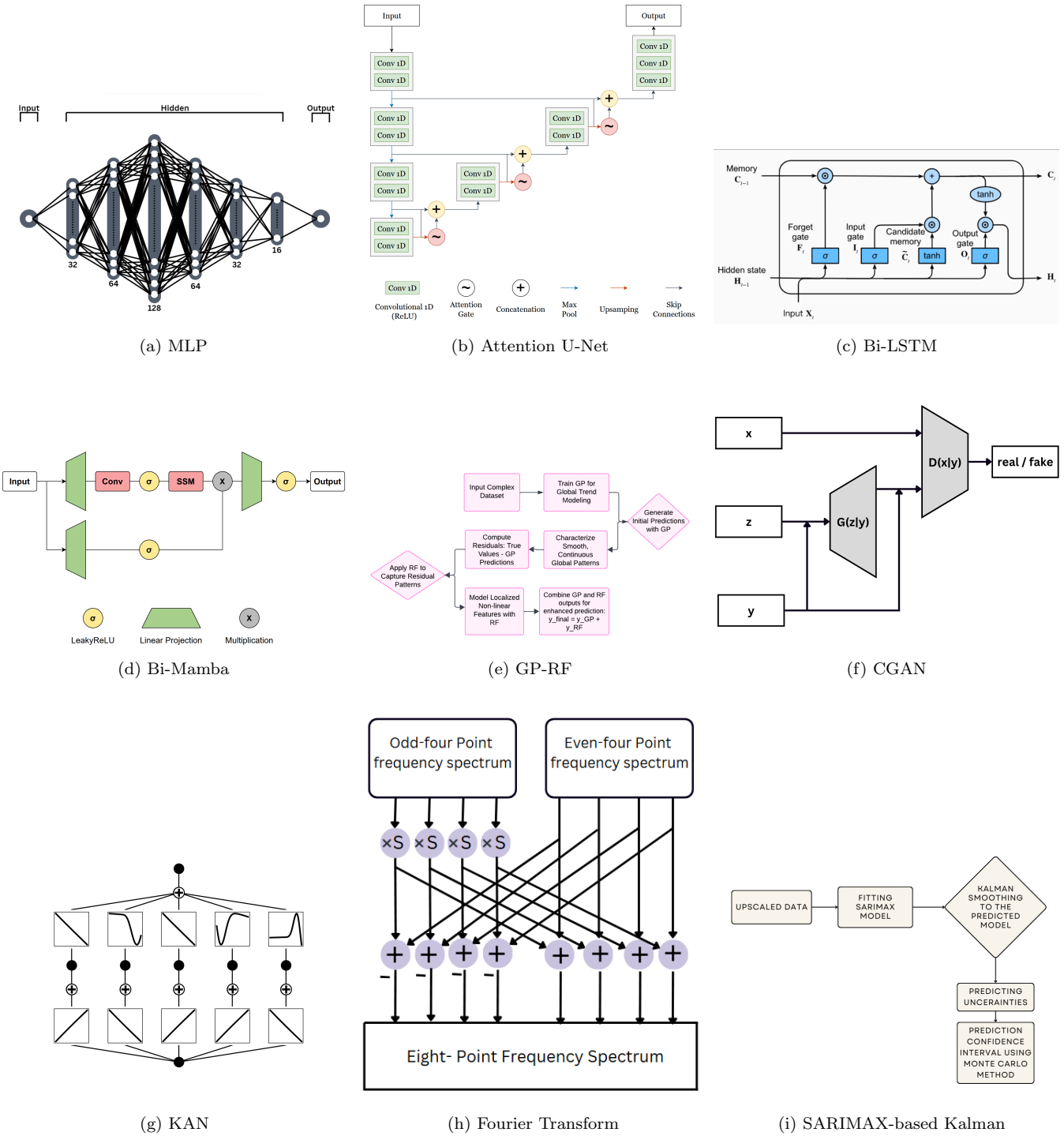
The output layer gives the prediction as follows:

$$\log_{10}(\text{flux})_{\text{predicted}} = W^{(\text{out})} \cdot h^{(l-1)} + b^{(\text{out})}. \tag{13}$$

The model is trained using the `Adam` optimizer (D. P. Kingma & J. Ba 2014) with 0.001 as the learning rate and MSE as the loss function. The number of epochs is set to 5000. The model was implemented using the TensorFlow framework with `Keras`.

Next, we perform the fivefold CV for all 521 GRBs individually and then report the average mean squared error (MSE) across all GRBs as our performance metric in Table 5.

The predicted GRB LCs given by the MLP model would be smooth and unrealistic. To bring about the inherently noisy nature of the logarithmic values of the natural flux values, Gaussian noise was added to each $\log_{10}$(flux) prediction in the same manner as in Section 2.3.



(a) MLP (b) Attention U-Net (c) Bi-LSTM

(d) Bi-Mamba (e) GP-RF (f) CGAN

(g) KAN (h) Fourier Transform (i) SARIMAX-based Kalman

Figure 3. Architecture of nine models shown from top left to bottom right in each row: (a) MLP; (b) Attention U-Ne; (c) Bi-LSTM; (d) Bi-MAMBA; (e) GP-RF; (f) CGAN; (g) KAN; (h) Fourier transform; and (i) SARIMAX-based Kalman. Architectures of Bi-Mamba, MLP, KAN, and Attention U-Net are actual representations of the models utilized.

2.3.3. Attention U-Net

U-Net is a convolutional neural network architecture designed primarily for biomedical image segmentation (O. Ronneberger et al. 2015) but has been successfully adapted for one-dimensional data such as time series (M. Perslev et al. 2019). The architecture consists of a symmetric encoder-decoder structure, where the encoder compresses the input into a latent representation, and the decoder reconstructs the output from this latent space. Latent space is a lower-dimensional representation of the data where only the key features are retained and the rest discarded. *Skip connections* between the encoder and decoder ensure the preservation of spatial details, which are critical for high-resolution reconstruction tasks.

For the LCR, we enhance the standard U-Net with an attention mechanism (O. Oktay et al. 2018), which is a section

of a neural network that enhances feature extraction and boosts performance on tasks involving complicated or sequential data by dynamically focusing on the most pertinent portions of the input data and giving distinct aspects varied degrees of priority, as stated in A. Vaswani et al. (2017). The attention mechanism enables the model to focus on relevant features at each resolution level by weighting them based on their importance. This allows the network to capture both global and local structures effectively.

In our implementation of Attention U-Net for GRB LCR, we employ fivefold CV to ensure robust performance evaluation as explained at the beginning of Section 2.3. To handle the inherent variability in LC lengths, all LCs are interpolated to a fixed length of 100 using uniform linear interpolation prior to training. This preprocessing ensures compatibility with the U-Net architecture, which requires fixed-size input tensors, without altering the temporal characteristics of the original LCs.

The attention mechanism is introduced through an *attention block*, which is integrated into the decoder path of the U-Net. Each attention block computes the importance weights as follows:

$$\psi_f = \sigma(\text{Conv1D}(f(\theta_x + \phi_g))), \tag{14}$$

where θ_x represents the feature map from the encoder, ϕ_g is the corresponding upsampled feature map from the decoder, f applies the ReLU activation function, and σ is the *sigmoid* activation function (explained in Table 1). The calculated weights ψ_f are then used to modulate the encoder features:

$$\text{attention output} = x \cdot \psi_f. \tag{15}$$

where x and g represent the encoder and decoder features, respectively.

The encoder-decoder structure of the model forms the backbone of the architecture. The encoder's hierarchical features are extracted from the input LC data using convolutional layers with ReLU activation and *He uniform* initialization. Mathematically, the output of each convolutional layer is given by

$$h^{(l)}(t) = \text{ReLU}(W^{(l)}.h^{(l-1)}(t) + b^{(l)}), \tag{16}$$

where $h^{(l)}(t)$ represents the output of the lth layer, $W^{(l)}$ and $b^{(l)}$ are weight and bias matrices, and t denotes the temporal index. Max-pooling operations reduce the temporal resolution at each stage, enabling the model to capture long-range temporal dependencies.

We implemented a 1D U-Net with an attention mechanism. The network consisted of three downsampling blocks with ConvID (kernel size = 3, padding = same, activation = ReLU), which reduces the sequence length (temporal dimension) by a factor of 2, followed by a bottleneck. The bottleneck is the sequence with the shortest length. This stage captures the most compressed and high-level features of the input sequence, allowing the model to learn global patterns across the time series. Then, we use three upsampling blocks with the *UpSampling1D* function to double the sequence length back to the original state. Each convolutional block used *He uniform* initialization. Attention blocks were inserted in the decoder path for feature refinement. These blocks help the network to selectively focus on the most relevant temporal features, enhancing the reconstruction quality. The model was trained using the `Adam` optimizer (learning rate = 0.001), with a batch size of 64 and for 1000 epochs. The input was log-transformed time values, and the output was the corresponding log-transformed flux values. Fivefold CV was used for evaluation, and final predictions were made using the full training set. The implementation of the Attention U-Net architecture was carried out using `TensorFlow` and `Keras`.

The data is transformed into a high-dimensional latent representation using two Conv1D layers with 256 filters, where 1D indicates that the convolution operates along a single spatial dimension (time, in our case), and filters refer to learnable kernels that extract features from the input LC data. The model is represented in Figure 3(b). This representation captures the most abstract features of the input sequence, which are essential for accurate reconstruction.

The decoder reconstructs the temporal sequence by progressively restoring the temporal resolution through upsampling layers, which increases the resolution of the input data. Skip connections between the encoder and decoder layers ensure that fine-grained temporal details are preserved. To enhance the utility of skip connections, attention blocks are applied (O. Oktay et al. 2018).

The model's last output layer is the Conv1D layer, which previously maps the reconstructed features to the previously anticipated flux values. It has a single filter. The complete model is trained to reduce the MSE between the observed and predicted logarithmic flux values. The loss function is written as

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\log_{10}(\hat{y}_i) - \log_{10}(y_i))^2\,, \tag{17}$$

where N is the number of data points, $\hat{y}_i$ represents the predicted flux, and y_i represents the observed flux.

Attention U-Net is well suited for tasks involving complex temporal structures like GRB LCs, where capturing long-range dependencies and local variations is critical (M. Perslev et al. 2019).

2.3.4. Long Short-term Memory Neural Network

Long short-term memory (LSTM; S. Hochreiter & J. Schmidhuber 1997) is a form of recurrent neural network (L. R. Medsker et al. 2001) that performs well in sequence prediction through its ability to capture long-term dependencies. Unlike traditional neural networks, LSTM incorporates feedback connections, allowing it to process data sequences rather than individual data points. This distinctive architecture equips LSTM with the ability to effectively learn and retain crucial information over time, making it a competent deep learning tool well suited for applications in speech recognition, language translation, time-series forecasting, and more (J. Cheng et al. 2016; Y. Zhang et al. 2016; Y. Hua et al. 2019).

The main feature of an LSTM is its memory cell, which is managed by three essential gates: the Input gate, the forget gate, and the output gate. These gates determine which information is added, removed, and output from the memory cell to learn long-range relationships within the data effectively.

In an LSTM network, a hidden state functions as the short-term memory, capturing information across time steps. This

Table 2
List of Variables and Symbols Used in the Bi-LSTM Model

Symbol	Variable
t	Time
x_t'	Input sequence
h_t'	Hidden state
h_{t-1}'	Previous hidden state
C_t	Cell state
C_{t-1}'	Previous cell state
f_t'	Forget gate
i_t'	Input gate
o_t'	Output gate
W_f'	Forget gate weights
W_i'	Input gate weights
W_o'	Output gate weights
b_f'	Forget gate bias
b_i'	Input gate bias
b_o'	Output gate bias

combination allows the LSTM to effectively maintain context and adjust its memory in response to new inputs, facilitating its ability to learn complex patterns across long sequences.

LSTM networks have proven valuable in astrophysics, particularly for time-series data analysis of dynamic celestial phenomena. An example is the LSTM-FCNN model, which combines LSTM and a fully connected neural network to measure time delays in strongly lensed SNe Ia, crucial for measuring the Hubble constant (S. Huber & S. Suyu 2024). Furthermore, LSTMs have been applied for solar flare prediction from active regions (H. Liu et al. 2019), effectively predicting them using time-series analysis. These examples show that LSTMs can outperform traditional methods in certain astrophysical time-series tasks.

Our model uses Bi-LSTM (A. Graves et al. 2005) to reconstruct the GRB LCs. The Bi-LSTM architecture extends the standard LSTM by processing the input data sequence in both forward and reverse directions, allowing it to capture temporal dependencies more effectively. This is particularly useful for GRB LCs, which feature complex temporal patterns such as flares and breaks. Bi-LSTMs also excel at handling missing data by learning from surrounding observations, enabling accurate imputation. The basic structure of the LSTM is described below and shown in Figure 3(c). Table 2 defines all the variables and symbols.

1. *Input gate.* It controls which new information should be incorporated into the cell state. The input is denoted with i. It computes a gating signal (i_t') using a sigmoid activation function applied to the weighted combination of current input (x_i') and the previously hidden stage (h_{t-1}'), where t denotes the time, along with a trainable weight (W_i') and bias (b_i'):
$$i_t' = \sigma(W_i' \cdot [h_{t-1}', x_t'] + b_i'). \quad (18)$$
2. *Forget gate.* It determines whether the amount of the previous memory cell ($C_{t-1}^{'}$) should be discarded or retained. The forget gate is denoted with f. This is also computed using a sigmoid activation function. The gate output is between 0 and 1, where a value close to 0 indicates that the information must be forgotten:
$$f_t' = \sigma(W_f' \cdot [h_{t-1}', x_t'] + b_f'). \quad (19)$$
3. *Cell state update.* It combines the retained information from the previous cell state and the new candidate memory. This update is governed by both the input and forget gates. The forget gate determines how much of the previous memory ($C_{(t-1)}'$) is preserved, while the input gate controls the amount of new information to be added:
$$C_t' = f_t' \cdot C_{t-1}' + i_t' \cdot \tilde{C}_t'. \quad (20)$$
4. *Output gate.* It specifies the portion of the updated cell state to be passed as the hidden state at the current time step. This is also computed using the sigmoid activation function:
$$o_t' = \sigma(W_o' \cdot [h_{t-1}', x_t'] + b_o'), \quad (21)$$
where o' denotes the output gate.

Finally, the updated cell state is passed through the hyperbolic tangent function, and the final hidden state is computed as
$$h_t' = o_t' \cdot \tanh C_t'. \quad (22)$$

In Bi-LSTM, the input sequence is processed in forward and reverse directions. The forward LSTM processes the set from the first to the last time step, learning dependencies from the past, while the backward LSTM processes the set from the end, retrieving information from future time steps. Each LSTM layer independently computes the gating signals, and hidden and cell states, as described earlier. The outputs from the forward and backward LSTMs are then merged at each time instance to form the final output:
$$h_t'^{\text{Bi-LSTM}} = [h_t'^{\text{forward}}, h_t'^{\text{backward}}]. \quad (23)$$

For hyperparameter optimization, *Optuna* was employed to fine-tune essential parameters, including the number of Bi-LSTM layers and the number of hidden units. The architecture consists of Bi-LSTM layers to capture both forward and backward temporal dependencies. In our model, we use four layers of Bi-LSTM, each with 100 hidden units, followed by a dense layer with Leaky Rectified Linear Unit (*LeakyReLU*) activation to help the LSTM model learn complex features of flares and breaks in the LC. *He uniform* initialization was applied to initialize the network weights, promoting stable convergence. *ReduceLROnPlateau*, a callback function in `Keras`, is used to reduce the learning rate when the validation loss plateaus to optimize the learning process further. The learning rate is set to the default value of 0.001, and the number of training epochs is fixed at 100. Once the optimal hyperparameters were determined, we performed another fivefold CV using these optimized settings to obtain the final

performance, reported as the train and test MSE in Table 5. The time step is taken as 1. The model was developed and trained utilizing the `TensorFlow` and `Keras` libraries.

2.3.5. Bi-Mamba Model

Mamba (A. Gu & T. Dao 2023) is a novel sequence modeling architecture based on state-space models, designed to efficiently capture long-range dependencies while reducing computational complexity. Unlike LSTM networks, which rely on explicit gating mechanisms to regulate memory updates, Mamba employs a structured state-space framework that enables selective long-range memory retention. This structured recurrence mechanism allows Mamba to model temporal dependencies effectively, achieving linear time complexity, making it particularly suitable for large-scale astronomical data sets such as GRB LCs, which exhibit intricate temporal variations.

The fundamental operation of Mamba is governed by its continuous-time state-space representation, where the hidden state s_t evolves according to the equation

$$s_t = As_{t-1} + Bx_t. \tag{24}$$

Here, $A \in \mathbb{R}^{N\times N}$ and $B \in \mathbb{R}^{1\times N}$ are learnable matrices, controlling the state transition dynamics and scaling the contribution of the input x_t, respectively. The final output at each time step is obtained using a learned transformation matrix $C \in \mathbb{R}^{1\times N}$:

$$y_t = Cs_t. \tag{25}$$

Since GRB LCs are discrete time-series data, Mamba is adapted to such inputs using a zero-order hold method—as described in G. Pechlivanidou & N. Karampetakis (2022) and A. Gu & T. Dao (2023)—which converts discrete inputs into piecewise constant signals. The resulting discrete-time formulation is given by

$$h_t = \bar{A}h_{t-1} + \bar{B}x_t, \tag{26}$$

$$y_t = \bar{C}h_t, \tag{27}$$

where $\bar{A}$, $\bar{B}$, and $\bar{C}$ are the discrete counterparts of the continuous model parameters.

To enhance temporal context modeling in GRB LC reconstruction, the Bi-Mamba model is used. Similar to Bi-LSTM, Bi-Mamba processes input sequences in both forward and reverse directions, ensuring that each time step benefits from contextual information from both past and future observations. This particularly benefits GRB LC reconstruction, where missing data segments require contextual information from surrounding observations. Bi-Mamba consists of two parallel Mamba models: a forward Mamba that processes the sequence in a standard temporal order and a backward Mamba that processes it in reverse. The outputs of both directions are concatenated at each time step to form the final bidirectional representation:

$$h_t'^{\text{Bi-Mamba}} = [h_t'^{\text{forward}}, h_t'^{\text{backward}}]. \tag{28}$$

Each directional Mamba model employs a selective memory retention mechanism, filtering out irrelevant noise while preserving meaningful historical context. Unlike LSTMs, which rely on discrete gating operations, Mamba's continuous-time state transition model enables a more flexible representation of temporal dependencies, particularly beneficial for modeling the complex structure of GRB LCs, including flares and breaks.

We used a modified version of the Mamba model from A. F. Thielmann et al. (2024) (shown in Figure 3(d)), which contains an output activation layer with the LeakyReLU activation function and a linear mapping at the end. The model was built using the PyTorch framework with hyperparameter tuning using *Optuna*. The data (D) was scaled to a range of [0, 1] using *MinMaxScaler*, given in Equation (4). The hidden state dimension (d_state) was set to 32, while the hidden layer expansion factor (d_expand) was configured to 80. A dropout rate of 0.1 was applied to mitigate overfitting. Training was conducted over 120 epochs with a batch size of 64, using the `Adam` optimizer with a learning rate of 10^{-4}.

2.3.6. GP-RF Model

To overcome the challenges of traditional machine learning models in representing localized patterns within complex LC, we build a hybrid model that combines the strengths of GPs and RFs.

In this method, the GP serves as the base model that captures continuous, smooth trends. At the same time, the RF acts as an embedded model to model the intricate, nonlinear patterns (e.g., flares and breaks) in the LCs that the GP might overlook. By analyzing the residuals, defined as the deviation between the predictions by the GP model and the observed values, the RF effectively learns to model the flares left unaddressed by the GP, allowing it to complement the GP's global predictive capabilities with its localized expertise. This hybrid approach offers a reliable solution for high predictive accuracy and reliability. As the base model, GP provides a smooth curve, and RF, as the meta-model, decreases the MSE.

GP forms the foundation of the hybrid model. The GP can capture global, nonlinear relationships as a probabilistic, nonparametric model while providing smooth predictions and uncertainty. The GP was trained on the input data to generate initial predictions representing the broad underlying structure of the LCs. The GP model was optimized using five random restarts, controlled by the parameter *n_restarts_optimizer* = 5, to improve the chances of finding the global optimum and reduce the likelihood of the model converging to local minima. These initial GP predictions serve as the baseline for the subsequent hybrid model, upon which further refinements and corrections are made.

Although the GP effectively models global trends, it exhibits limitations in capturing finer, localized variations in the data. To address this, the residuals, defined as the difference between the observed values and the GP predictions, were calculated:

$$\text{residuals} = y_{\text{observed}} - y_{\text{predicted}}. \tag{29}$$

Here, y_{observed} is the actual flux values corresponding to the data, and $y_{\text{predicted}}$ is the flux values predicted by the GP model.

These residuals were then modeled using RFs, a powerful ensemble method known for its ability to handle nonlinear relationships and complex interactions. RFs were proposed by Leo Breiman (L. Breiman 2001) to build a predictor ensemble with a set of decision trees (L. Breiman et al. 2017) growing in randomly selected data subspaces. The model is shown in Figure 3(e).

The RF component was optimized through a systematic hyperparameter tuning process using *RandomizedSearchCV*

with 10 iterations and threefold CV. The hyperparameters were carefully defined to balance model complexity and computational efficiency, encompassing key parameters such as the number of estimators (ranging from 100 to 200 trees), maximum tree depth (10, 20, or unrestricted), minimum samples required for node splitting (2 or 5), and minimum samples per leaf node (1 or 2). This comprehensive parameter optimization strategy ensures reliable model performance while maintaining adaptability to local features in the LCs. The model was implemented using the `scikit-learn` library.

The final prediction of the hybrid model is obtained by combining the outputs of the GP and the RF:

$$y_{\text{final}} = y_{\text{GP}} + y_{\text{RF}}. \tag{30}$$

2.3.7. CGAN

I. J. Goodfellow et al. (2014) provide an innovative approach to reconstruct the LCs. Unlike traditional models that rely on Markov Chains—where the current state is dependent on the previous one—CGANs bypass this sequential dependency during the learning phase (M. Mirza & S. Osindero 2014), incorporating it only during gradient backpropagation. These networks are referred to as generative models and are particularly useful when dealing with complex data distributions, reducing overfitting by incorporating a dual-model approach.

These adversarial networks have a generator model $G(z)$ and a discriminator model $D(x)$ working harmoniously. Both the generator and the discriminator try to improve their results. If the generator constructs more real samples x from random noise z, it becomes challenging for the discriminator to distinguish between the actual data and the synthetic data. Their adversarial relationship is mathematically described as

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim q(x)}[\log D(x)] + \mathbb{E}_{z\sim r(z)}[\log(1 - D(G(z)))], \tag{31}$$

where the $\mathbb{E}_{x\sim q(x)}[\log D(x)]$ term denotes the expected value, $\mathbb{E}$, of $\log D(x)$, where x is sampled from the observed data distribution $q(x)$. Likewise, $\mathbb{E}_{z\sim r(z)}[\log(1 - D(G(z)))]$ is the expected value of $\log(1 - D(G(z)))$, where z is sampled from the input noise distribution $r(z)$ of the generator $G(z)$.

M. Mirza & S. Osindero (2014) extend the CGAN architecture by incorporating a condition set y into the generator and discriminator, guiding the data generation process. The architecture of CGAN (as seen in Figure 3(f)) comprises a generator $G(z|y)$ with two inputs: a condition set y and a random noise vector z, and a conditioned discriminator $D(x|y)$ on y. The generator aims to create new samples that closely resemble the original samples by using the noise and the condition set as input. After the generation of samples, these, along with the real samples, are fed into the discriminator. The objective for the discriminator is to effectively classify between the generated and the real samples, retrospectively training the generator and discriminator in the process. It is mathematically represented as

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim q(x)}[\log D(x|y)] + \mathbb{E}_{z\sim r(z)}[\log(1 - D(G(z|y)))]. \tag{32}$$

In our implementation, the generator begins with a dense layer containing 256 neurons and the LeakyReLU activation function (B. Xu et al. 2015). It includes a positive slope m and comparatively a tiny negative slope αm, preventing the occurrences of dead neurons. In our case, α is taken to be 1×10^{-2}. Following the dense layer, we utilized two convolution-1D layers with 64 and 16 filter sizes, respectively. Paired with ReLU activation, these convolution-1D layers enhance the feature extraction process, ensuring distinct, high-quality features. The flux and time values are preprocessed by converting them to a symmetric logarithmic scale. The time values act as condition sets for the CGAN, the flux values provide the actual data, and the noise set contains random uniform noise samples from 0 to 1 covering the whole domain. All layers in the discriminator, except the final binary classification layer, mirror those of the generator. This design ensures a balanced model complexity, preventing either network from overpowering the other. The discriminator loss is computed as the sum of binary cross-entropy losses for correctly classifying real samples as real and generated samples as fake. The generator loss is defined as the binary cross-entropy between the discriminator's predictions on generated samples and the target label for real samples, thereby encouraging the generator to produce more realistic outputs. This results in reliable data-augmentation capabilities for the reconstruction of time-series data (R. Fu et al. 2019; E. Brophy et al. 2023).

The proposed method uses the `TensorFlow` and `Keras` libraries for its implementation, with 5000 epochs and a batch size of 256. The `Adam` optimizer was used to train the model, optimizing the loss function at a learning rate of 1×10^{-4}.

2.3.8. KANs

The KAN model was first introduced by Z. Liu et al. (2024) and has since been studied extensively. One of its key advantages is its ability to recognize complex patterns and relationships in data.

A distinguishing feature of KANs is their activation mechanism. Instead of applying activation functions at the nodes (as in standard neural networks), KANs place nonlinear functions along the connections between nodes. These functions are composed of adjustable splines and basis functions, allowing for finer control over how information flows through the network.

One of the most significant advantages of KANs in scientific contexts is their interpretability. Because each learnable function is univariate and localized, it can be visualized and analyzed directly, offering insights into each input variable's individual contribution, often lost in traditional black-box neural networks. This makes KANs well suited for analyzing sparse or structured data sets, such as those encountered in GRB LC reconstructions.

Pros:

1. Greater interpretability through visual inspection of learned functions.

2. Adaptive flexibility due to spline-based transformations.
3. Potential for better generalization on structured or low-sample-size data.

Cons:

1. Increased computational complexity in training, especially in high dimensions.
2. Tooling and ecosystem are new compared to MLPs and CNNs, and lack support and reliability.
3. High sensitivity to hyperparameters, particularly regarding the number and placement of spline knots.

However, many of these limitations have been mitigated, mainly in our case, due to the one-dimensional nature of the data. Furthermore, we employed *Optuna*-based hyperparameter tuning to ensure the selection of optimal parameters for model efficiency. The implementation was rigorously validated within the PyTorch framework to ensure correctness and stability.

Let the input layer consist of n_0 nodes, and the lth layer have n_l nodes. For any node j in the $(l+1)$th layer, the activation $x_{l+1,j}$ is computed as

$$x_{l+1,j} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}), \tag{33}$$

where $\phi_{l,j,i}(\cdot)$ denotes the learnable univariate function connecting node i of layer l to node j of layer $l+1$.

This approach allows KANs to capture intricate data patterns with high precision, making them well suited for reconstructing missing or incomplete flux data in GRB observations.

The overall computation for a KAN with l layers is expressed as

$$\mathrm{KAN}(x) = (\Phi_{l-1} \circ \Phi_{l-2} \circ \cdots \circ \Phi_0)(x), \tag{34}$$

where Φ_l represents the activation function matrix for the lth layer. This structure supports deeper networks by stacking layers, analogous to how depth enhances expressiveness in traditional neural networks. In our case, we have used only a single layer of neural network in KAN.

To optimize the spline-based activation functions, each univariate function is represented as a combination of basis splines as

$$\phi(x) = w_b b(x) + w_s \,\mathrm{spline}(x). \tag{35}$$

The basis function $b(x)$ is expressed as

$$b(x) = \mathrm{silu}(x) = \frac{x}{(1 + e^{-x})}, \tag{36}$$

$$\mathrm{spline}(x) = \sum_i c_i B_i(x), \tag{37}$$

with c_i as the trainable coefficient that scales the contribution of the ith basis function and B_i is the ith B-spline basis function. This parameterization ensures smooth and adaptable transformations while maintaining computational efficiency.

We construct the KAN model with a width configuration of [1, 5, 1], indicating that the input and output each consist of a single feature vector (or column), while the hidden layer contains five neurons. The model also utilizes a grid size of 5 (shown in Figure 3(g)).

The training process employs the `Adam` optimizer, with a learning rate of 0.001 and a batch size of 64, trained on 350 epochs. The model's performance is monitored every five epochs, and the best loss value is tracked throughout the training process.

2.4. The Statistical Approach

These models follow the same preprocessing and result calculation steps as mentioned in Section 2.3.

2.4.1. Fourier Transform

Transformation in mathematics is a technique to transform or map a function from its original function space to a different function space. The Fourier transform is, therefore, a transformation technique that maps the functions dependent on the time domain into functions dependent on the temporal frequency domain. This breaks down a signal into a sum of sinusoidal functions. The Fourier transform of a function $f(t)$ is defined by $\hat{f}(\omega)$, which represents the frequency domain representation of the signal

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt. \tag{38}$$

Fourier transform helps to understand the data's periodicity and reduce noise.

This concept was implemented to reconstruct GRB LCs by transforming time-domain flux values into frequency domain coefficients. The initial data was first processed using fast Fourier transform (FFT; U. Oberst 2007) on the flux values. The interpolation technique imputes the new flux values, and the uncertainties were predicted using the normal distribution. This process helps us understand how flux values may vary over time, taking uncertainties into account. Figure 3(h) is the FFT synthesis flow diagram that shows how the FFT works. It shows how the FFT combines two four-point spectra into an eight-point spectrum. The "$\times S$" symbol represents the signal multiplied by a sinusoid of appropriate frequency.

Implementing the Fourier transform helps deal with noisy and sparse data. This method also aids in understanding the periodic patterns and gives a cleaner representation of the underlying signal by removing unwanted noise. Furthermore, Gaussian smoothing and interpolation techniques enhance the model's performance, increasing its reliability.

2.5. SARIMAX-based Kalman Smoothing Model

We researched the Seasonal Autoregressive Integrated Moving Average with Exogenous Regressors (SARIMAX; F. R. Alharbi & D. Csala 2022) and Kalman smoothing (A. Aravkin et al. 2013), and how integrating both models helps reconstruct the GRB LCs. This combination is quite effective since it combines the statistical capability of SARIMAX for time-series modeling with the refinement capabilities of Kalman smoothing, which considers the entire data for the smoothing process. The Kalman smoothing in our implementation is applied through the state-space representation provided by the `statsmodels` package. The time and flux data are preprocessed by transforming them into the logarithmic domain to stabilize variance and enhance the relationship between time and flux. This ensures computational stability.

The SARIMAX model captures the observed flux data's temporal dependencies, trends, and noise. An ARIMA model (S. Ho & M. Xie 1998) of order (p, d, q) is tuned accordingly

Table 3
Comparison of Various Attributes of the Models

Attributes	GP	Bi-Mamba	MLP	Fourier	GP-RF	Bi-LSTM	CGAN	SARIMAX	KAN	U-Net
High throughput	✓	⋯	✓	✓	⋯	⋯	⋯	⋯	⋯	⋯
Good sparse data performance	✓	⋯	⋯	⋯	✓	⋯	⋯	✓	✓	⋯
Long-term memory	✓	✓	⋯	⋯	✓	✓	⋯	⋯	⋯	⋯
Low computational cost	✓	⋯	✓	✓	⋯	⋯	⋯	✓	⋯	⋯
Resilience to noise overfitting	✓	⋯	⋯	✓	✓	✓	✓	⋯	⋯	✓
Captures complex patterns	⋯	✓	✓	⋯	⋯	✓	✓	✓	✓	✓
Captures periodic components	⋯	⋯	⋯	✓	⋯	⋯	⋯	✓	⋯	⋯
Models' data distribution	✓	✓	✓	⋯	✓	✓	✓	⋯	✓	✓

to model the LC's temporal dynamics. Here, p is the number of autoregressive terms, d is the number of nonseasonal differences, and q is the number of lagged forecast errors in the prediction equation. An order of (1, 1, 1) was used that often captures basic temporal structure with fewer parameters, avoiding overfitting while reducing autocorrelation in residuals. In our model, we applied Kalman smoothing to refine the SARIMAX output. Kalman smoothing works in such a way that it estimates the system's state using measurements from both past and future time steps. Let $\boldsymbol{x}_t$ denote the state vector at time t, modeled as

$$\boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{w}_t, \tag{39}$$

where $\boldsymbol{A}$ is the state transition matrix, $\boldsymbol{B}$ is the control input matrix, $\boldsymbol{u}_t$ is the control input, and $\boldsymbol{w}_t$ represents process noise. In this work, we utilize the SARIMAX framework without incorporating exogenous regressors; hence, the control input term $\boldsymbol{B}\boldsymbol{u}_t$ is omitted from our practical implementation.

The observation $\boldsymbol{y}_t$ is modeled as

$$\boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{v}_t, \tag{40}$$

where $\boldsymbol{C}$ is the observation matrix and $\boldsymbol{v}_t$ represents measurement noise. However, in our implementation, we have added noise, as mentioned at the start of Section 2.3.

The Kalman smoothing algorithm minimizes the MSE of state estimates, yielding a refined sequence of flux values $\hat{\boldsymbol{F}}_t$. After Kalman smoothing, uncertainties were added (as seen in the model flow Figure 3(i)).

The model was implemented using Python libraries such as `statsmodels`, specifically the SARIMAX class and `Matplotlib` for visualizing the final results, including the time-series plots and confidence bands. The SARIMAX fitting is optimized for computational efficiency, and specifically, the Kalman smoother estimated the latent states (*smoothed_state [0]*) to refine noisy measurements by incorporating information from both past and future time steps. These smoothed flux values were then interpolated across a dense logarithmic time grid to reconstruct the LC. The `SciPy` library's *Gaussian_filter1d* function was applied to the predicted confidence interval bounds.

3. Attributes of the Models

The attributes of each model, as shown in Table 3, provide insight into their performance. Using these models, the following inferences can be made:

1. *Bi-Mamba.* Best for sequential GRB data with long-term dependencies. It adapts to different data distributions but requires careful management of computational cost and noise overfitting.
2. *MLP.* Flexible across data distributions but prone to noise overfitting without regularization. Less suitable for sparse, irregular, or sequential GRB data.
3. *Fourier transform.* Well suited for periodic or smooth components in GRB signals, offering efficiency in high-throughput tasks but less effective for nonperiodic or nonlinear data.
4. *GP-RF.* Effective for sparse or noisy GRB data, combining probabilistic modeling and reliable feature inference. High computational cost for large data sets, but it handles complex data distributions well.
5. *Bi-LSTM.* Highly effective for sequential data due to their ability to retain long-term dependencies and capture nonlinear relationships within the data. However, they are less effective when the data is sparse.
6. *CGAN.* Excels at capturing complex, nonlinear relationships and learning high-dimensional data distributions. However, it can be computationally expensive and require careful training.
7. *SARIMAX-based Kalman model.* Efficient in handling sparse data and are computationally efficient, making them suitable for data sets with limited information. They effectively capture nonlinear and periodic relationships, making them ideal for modeling time-series data.
8. *KAN.* Excellent for capturing complex nonlinear patterns, though the model requires sufficient training data and careful tuning to avoid noise overfitting.
9. *Attention U-Net.* Effective for GRB data with complex patterns due to its ability to focus on critical features through an attention mechanism. However, significant computational resources and careful tuning of attention parameters are required to avoid overfitting.

These proposed models are detailed in Table 3.

4. Results

Decreasing the uncertainty associated with the W07 parameters is an objective of LCR. To evaluate this, the error fractions (EFs) are calculated for every model parameter in both the primary data sets and after reconstructing them using the model. The EFs for the three W07 parameters as described in M. G. Dainotti et al. (2023b) are represented in

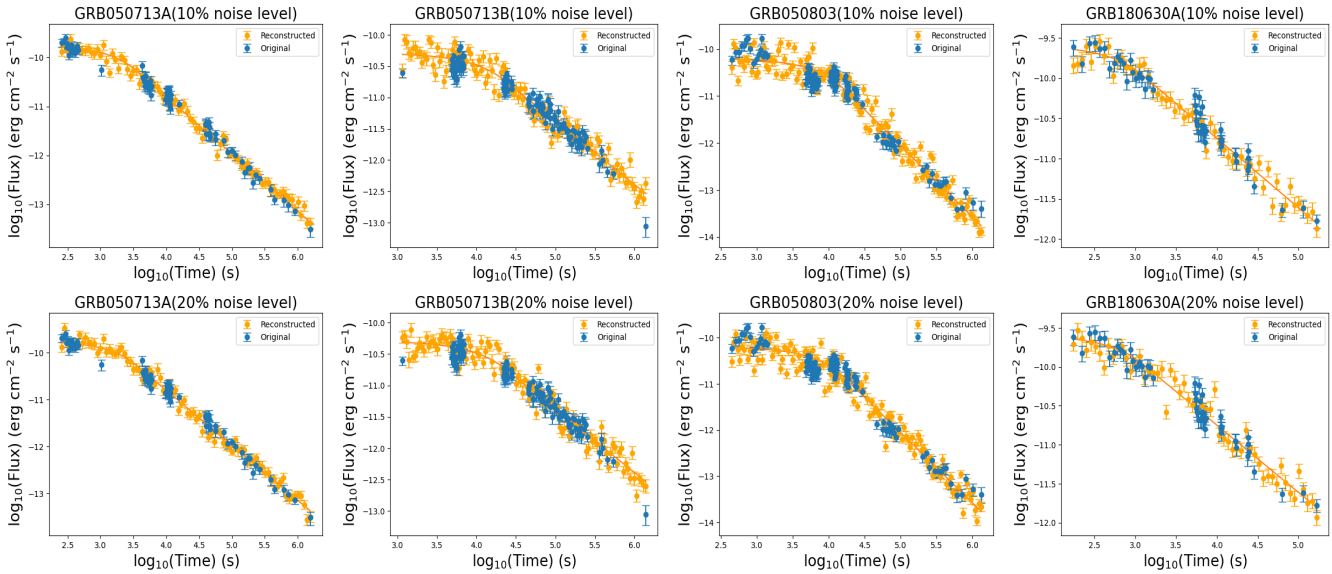


Figure 4. Reconstruction of LCs using the W07 function. Row 1 shows the reconstructed LCs at 10% noise level and row 2 shows the reconstructed LCs at 20% noise level for all four classes: (i) good GRBs (column 1); (ii) a break toward the end of the GRB LC (column 2); (iii) flares or bumps in the afterglow (column 3); and (iv) flares or bumps with a double break toward the end of the LC (column 4).

Equations (41), (42), and (43):

$$\mathrm{EF}_{\log_{10}(T_a)} = \left| \frac{\Delta \log_{10}(T_a)}{\log_{10}(T_a)} \right|, \tag{41}$$

$$\mathrm{EF}_{\log_{10}(F_a)} = \left| \frac{\Delta \log_{10}(F_a)}{\log_{10}(F_a)} \right|, \tag{42}$$

$$\mathrm{EF}_{\alpha_a} = \left| \frac{\Delta \alpha_a}{\alpha_a} \right|. \tag{43}$$

We compute the reduction in the percentage of EF to evaluate the enhancement in fit post-reconstruction.

$$\%_{\mathrm{DEC}} = \frac{|\mathrm{EF}_X^{\mathrm{after}}| - |\mathrm{EF}_X^{\mathrm{before}}|}{|\mathrm{EF}_X^{\mathrm{before}}|} \times 100. \tag{44}$$

4.1. Results from the Willingale Model

The reconstruction results for all four classes of GRBs under 10% and 20% noise levels are illustrated in Figure 4. For all classes of GRBs, the LCs are reconstructed to demonstrate how increasing noise levels affect the distribution of data points near the W07 fit. As anticipated, increasing the noise level generally leads to higher uncertainty due to the broader uncertainty around the best-fit line. However, in our case, the decrease in uncertainty does not follow this pattern with increasing noise levels. This is because we are considering the entire sample. The data points deviate significantly from the model function for GRBs exhibiting flares and breaks. As a result, with more significant noise levels, the decrease in uncertainty is less pronounced. Additionally, the EFs for the W07 model applied to 521 good GRBs at both 10% and 20% noise levels are summarized in Table 4. The results include the histogram distribution of the percentage decrease for all three W07 parameters at these noise levels.

A reduction of 24.5% for $\log T_a$, 25.7% for $\log F_a$, and 36.2% for α is observed at a 10% noise level. However, for the subset containing only 207 good GRBs, it performs the best, showcasing 33.3% for $\log T_a$, 35.0% for $\log F_a$% and 43.3% for α with no additional outliers. As expected, the reductions were marginally smaller, with 21.2% for $\log T_a$, 22.9% for $\log F_a$, and 34.7% for α, for a 20% noise level.

4.2. Results from the Machine Learning and Statistical Methods

The reconstruction for each category of GRBs for all models is shown in Figures 5 and 6, and the histogram distribution of the relative percentage decrease for the three W07 parameters, computed using the entire GRB sample, is illustrated in Figures 7 and 8. Table 4 shows the EF (before and after LCR) and the percentage reduction of EF for all models across the entire GRB sample. Along with these, we conducted a fivefold CV for all the models to calculate the train and test MSE, ensuring the reliability of our approach. Table 5 compares the performance of all the machine learning models.

5. Summary and Conclusion

We analyzed various unique approaches, showcasing the efficacy of reconstructing LCs. Below, we summarize the key findings of our results:

1. The W07 model (10% noise level) is most effective for reconstructing 207 good GRBs, achieving the highest uncertainty reduction across all parameters while eliminating outliers. However, its performance diminishes with flares or breaks in the data, limiting its applicability. The model fundamentally relies on the Willingale function, which is designed to represent a generalized LC structure. As such, it does not explicitly account for complex features like flares or breaks often observed in half of the GRBs. Similarly, the GP approach shows reduced reliability in handling GRBs with flares or breaks. Although it provides a more flexible nonparametric framework compared to the fixed functional form of W07, it still tends to follow the overall smooth trend of the LC and has issues in reconstructing transient deviations accurately.
2. The MLP model provides the lowest fivefold Test MSE of 0.0275 among deep learning models, making it the

Table 4
The W07 $\log T_a$, $\log F_a$, and α EFs before and after Reconstruction (with a Decrease in Relative Percentage in Error for All Three Parameters)

GRB ID (1)	$EF_{\log_{10}(T_i)}$ (2)	$EF_{\log_{10}(F_i)}$ (3)	EF_{α_i} (4)	$EF_{\log_{10}(T_i)}$ RC (5)	$EF_{\log_{10}(F_i)}$ RC (6)	EF_{α_i} RC (7)	$\%\log_{10}(T_i)$ (8)	$\%\log_{10}(F_i)$ (9)	$\%\alpha_i$
W07 10% Noise									
050315	0.00928	0.00285	0.0444	0.00894	0.00268	0.0431	−3.61	−6.05	−2.90
050318	0.0106	0.00554	0.0458	0.0108	0.00540	0.0417	1.78	−2.65	−9.11
050319	0.0166	0.00389	0.0400	0.0161	0.00370	0.0319	−2.89	−4.84	−20.2
050401	0.00907	0.00202	0.0265	0.00772	0.00174	0.0154	−14.9	−13.5	−41.8
050730	0.00439	0.00263	0.0225	0.00332	0.00205	0.0153	−24.5	−22.1	−32.1
050802	0.00896	0.00267	0.0226	0.00751	0.00239	0.0147	−16.1	−10.6	−35.2
050803	0.0113	0.00450	0.0320	0.0106	0.00419	0.0228	−6.38	−7.03	−28.9
W07 20% Noise									
050315	0.00928	0.00285	0.0444	0.00883	0.00267	0.0424	−4.78	−6.42	−4.40
050318	0.0106	0.00554	0.0458	0.0107	0.00562	0.0462	0.85	1.37	0.720
050319	0.0166	0.00389	0.0400	0.0161	0.00377	0.0325	−3.06	−3.24	−18.9
050401	0.00907	0.00202	0.0265	0.00779	0.00178	0.0156	−14.0	−11.7	−41.3
050730	0.00439	0.00263	0.0225	0.00332	0.00206	0.0154	−24.4	−21.9	−31.9
050802	0.00896	0.00267	0.0226	0.00786	0.00248	0.0151	−12.2	−7.32	−33.1
050803	0.0113	0.00450	0.0320	0.0113	0.00437	0.0245	−0.370	−3.04	−23.4
GP									
050315	0.00928	0.00285	0.0444	0.0105	0.00288	0.0403	13.3	0.940	−9.14
050318	0.0106	0.00554	0.0458	0.0105	0.00539	0.0428	−1.12	−2.79	−6.71
050319	0.0166	0.00389	0.0400	0.0124	0.00356	0.0355	−24.9	−8.49	−11.5
050401	0.00907	0.00202	0.0265	0.00671	0.00167	0.0149	−26.0	−17.1	−43.7
050730	0.00439	0.00263	0.0225	0.00326	0.00205	0.0152	−25.7	−22.2	−32.7
050802	0.00896	0.00267	0.0226	0.00856	0.00260	0.0156	−4.41	−2.81	−31.0
050803	0.0113	0.00450	0.0320	0.0106	0.00402	0.0239	−5.97	−10.7	−25.3
Bi-Mamba									
050315	0.00928	0.00285	0.0444	0.00826	0.00260	0.0400	−10.9	−8.76	−9.75
050318	0.0106	0.00554	0.0458	0.0103	0.00536	0.0443	−2.63	−3.26	−3.36
050319	0.0166	0.00389	0.0400	0.0123	0.00371	0.0387	−25.7	−4.74	−3.34
050401	0.00907	0.00202	0.0265	0.00683	0.00164	0.0146	−24.7	−18.5	−44.9
050730	0.00439	0.00263	0.0225	0.00325	0.00207	0.0152	−26.0	−21.5	−32.7
050802	0.00896	0.00267	0.0226	0.00771	0.00241	0.0147	−13.9	−9.90	−35.1
050803	0.0113	0.00450	0.0320	0.0102	0.00391	0.0226	−9.63	−13.1	−29.4
MLP Model									
050315	0.00928	0.00285	0.0444	0.00658	0.00202	0.0308	−29.1	−29.3	−30.7
050318	0.0106	0.00554	0.0458	0.00795	0.00402	0.0319	−25.0	−27.6	−30.3
050319	0.0166	0.00389	0.0400	0.00979	0.00262	0.0272	−40.9	−32.8	−32.0
050401	0.00907	0.00202	0.0265	0.00494	0.00121	0.0121	−45.6	−39.8	−54.4
050730	0.00439	0.00263	0.0225	0.00239	0.00151	0.0113	−45.6	−42.7	−49.7
050802	0.00896	0.00267	0.0226	0.00559	0.00175	0.0116	−37.6	−34.5	−48.5
050803	0.0113	0.00450	0.0320	0.00757	0.00295	0.0175	−33.1	−34.5	−45.4
Fourier Transform									
050315	0.00928	0.00285	0.0444	0.00940	0.00279	0.0420	1.29	−2.29	−5.27
050318	0.0106	0.00554	0.0458	0.0118	0.00600	0.0493	11.6	8.28	7.66
050319	0.0166	0.00389	0.0400	0.0105	0.00350	0.0389	−36.8	−10.0	−2.78
050401	0.00907	0.00202	0.0265	0.00698	0.00173	0.0155	−23.0	−14.2	−41.6
050730	0.00439	0.00263	0.0225	0.00332	0.00210	0.0155	−24.4	−20.2	−31.2
050802	0.00896	0.00267	0.0226	0.00803	0.00253	0.0154	−10.4	−5.48	−32.0
050803	0.0113	0.00450	0.0320	0.0108	0.00413	0.0244	−4.50	−8.25	−23.7
GP-RF Model									
050315	0.00928	0.00285	0.0444	0.00965	0.00272	0.0410	4.03	−4.60	−7.51
050318	0.0106	0.00554	0.0458	0.0103	0.00529	0.0432	−3.09	−4.56	−5.81
050319	0.0166	0.00389	0.0400	0.0117	0.00342	0.0380	−29.3	−12.0	−5.08
050401	0.00907	0.00202	0.0265	0.00681	0.00171	0.0152	−24.9	−15.1	−42.5

Table 4
(Continued)

GRB ID (1)	$EF_{\log_{10}(T_i)}$ (2)	$EF_{\log_{10}(F_i)}$ (3)	EF_{α_i} (4)	$EF_{\log_{10}(T_i)}$ RC (5)	$EF_{\log_{10}(F_i)}$ RC (6)	EF_{α_i} RC (7)	$\%\log_{10}(T_i)$ (8)	$\%\log_{10}(F_i)$ (9)	$\%\alpha_i$
050730	0.00439	0.00263	0.0225	0.00679	0.00416	0.0314	54.5	58.1	39.4
050802	0.00896	0.00267	0.0226	0.00830	0.00254	0.0154	−7.30	−5.02	−32.0
050803	0.0113	0.00450	0.0320	0.0108	0.00407	0.0238	−4.58	−9.69	−25.6
Bi-LSTM Model									
050315	0.00928	0.00285	0.0444	0.00814	0.00271	0.0414	−12.3	−5.04	−6.69
050318	0.0106	0.00554	0.0458	0.00996	0.00539	0.0443	−5.98	−2.84	−3.28
050319	0.0166	0.00389	0.0400	0.0102	0.00369	0.0410	−38.2	−5.26	2.52
050401	0.00907	0.00202	0.0265	0.00680	0.00166	0.0150	−25.0	−17.6	−43.3
050730	0.00439	0.00263	0.0225	0.00328	0.00204	0.0152	−25.4	−22.6	−32.7
050802	0.00896	0.00267	0.0226	0.00853	0.00259	0.0158	−4.77	−3.26	−30.1
050803	0.0113	0.00450	0.0320	0.0103	0.00394	0.0225	−9.13	−12.5	−29.6
CGAN Model									
050315	0.00928	0.00285	0.0444	0.00900	0.00283	0.0430	−3.05	−0.83	−3.18
050318	0.0106	0.00554	0.0458	0.0100	0.00508	0.0404	−5.24	−8.27	−11.9
050319	0.0166	0.00389	0.0400	0.00961	0.00353	0.0399	−41.9	−9.24	−0.280
050401	0.00907	0.00202	0.0265	0.00701	0.00171	0.0154	−22.7	−15.2	−41.8
050730	0.00439	0.00263	0.0225	0.00328	0.00208	0.0154	−25.4	−21.1	−31.7
050802	0.00896	0.00267	0.0226	0.00730	0.00238	0.0145	−18.5	−11.0	−35.7
050803	0.0113	0.00450	0.0320	0.0105	0.00412	0.0235	−6.99	−8.42	−26.6
SARIMAX+Kalman Model									
050315	0.00928	0.00285	0.0444	0.00845	0.00268	0.0417	−8.87	−6.01	−6.11
050318	0.0106	0.00554	0.0458	0.0112	0.00596	0.0472	5.36	7.46	3.02
050319	0.0166	0.00389	0.0400	0.0127	0.00361	0.0395	−23.6	−7.27	−1.30
050401	0.00907	0.00202	0.0265	0.00822	0.00188	0.0169	−9.39	−6.75	−36.2
050730	0.00439	0.00263	0.0225	0.00334	0.00208	0.0155	−24.1	−21.0	−31.3
050802	0.00896	0.00267	0.0226	0.00856	0.00262	0.0160	−4.48	−1.96	−29.2
050803	0.0113	0.00450	0.0320	0.0108	0.00408	0.0248	−4.50	−9.29	−22.5
KAN Model									
050315	0.00928	0.00285	0.0444	0.00810	0.00254	0.0391	−12.7	−10.9	−11.8
050318	0.0106	0.00554	0.0458	0.00970	0.00526	0.0456	−8.46	−5.18	−0.500
050319	0.0166	0.00389	0.0400	0.0135	0.00331	0.0286	−18.3	−14.9	−28.5
050401	0.00907	0.00202	0.0265	0.00695	0.00165	0.0149	−23.4	−18.0	−43.8
050730	0.00439	0.00263	0.0225	0.00330	0.00209	0.0153	−25.0	−20.9	−32.1
050802	0.00896	0.00267	0.0226	0.00730	0.00233	0.0145	−18.5	−12.8	−35.8
050803	0.0113	0.00450	0.0320	0.0102	0.00395	0.0222	−9.83	−12.2	−30.5
Attention U-Net									
050315	0.00928	0.00285	0.0444	0.00785	0.00214	0.0294	−15.4	−25.2	−33.8
050318	0.0106	0.00554	0.0458	0.00818	0.00417	0.0330	−22.8	−24.7	−28.0
050319	0.0166	0.00389	0.0400	0.00984	0.00275	0.0298	−40.6	−29.3	−25.5
050401	0.00907	0.00202	0.0265	0.00504	0.00123	0.0123	−44.4	−38.9	−53.5
050730	0.00439	0.00263	0.0225	0.00240	0.00149	0.0113	−45.4	−43.3	−49.9
050802	0.00896	0.00267	0.0226	0.00579	0.00179	0.0118	−35.4	−33.1	−47.7
050803	0.0113	0.00450	0.0320	0.00756	0.00295	0.0174	−33.1	−34.4	−45.6

Note. Columns (1), (2), and (3) provide EFs for the original W07 fit; columns (4), (5), and (6) show the EF for the W07 fit after LCs reconstruction. Columns (7), (8), and (9) show a percentage decrease in the EF after the reconstruction.

(This table is available in its entirety in machine-readable form in the online article.)

best option when prioritizing reconstruction accuracy. It also offers a high uncertainty reduction (37.2% for $\log T_a$, 38.0% for $\log F_a$, and 41.2% for α) while maintaining low outlier rates across all parameters.

3. The Attention U-Net model achieves a higher test MSE of 0.134 but excels in minimizing outliers, particularly for $\log T_a$ and $\log F_a$. It also exhibits the highest uncertainty reduction across all parameters (37.9% for $\log T_a$, 38.5% for $\log F_a$, and 41.4% for α).
4. In comparison, the Bi-LSTM and Bi-Mamba models show comparable levels of uncertainty reduction across the three parameters: 21.2%, 21.5%, and 26.1% for Bi-LSTM, and 20.8%, 21.2%, and 27.6% for Bi-Mamba, respectively. They are an effective alternative, especially

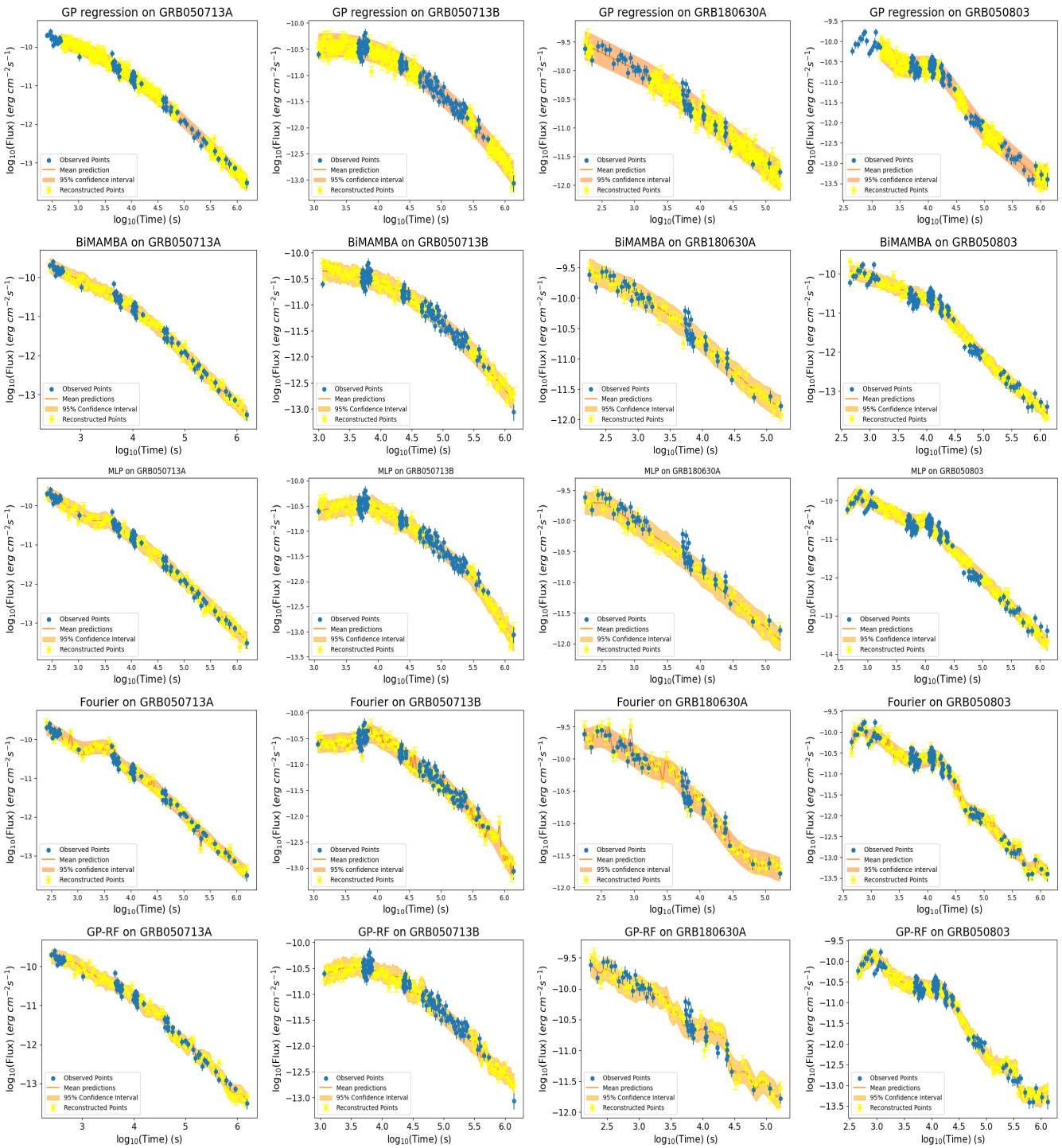


Figure 5. Reconstruction of LCs for all four varieties of GRBs are shown in a grid with four types of GRBs (left to right): (i) good GRBs (column 1); (ii) a GRB LC with a break toward the end (column 2); (iii) flares or bumps in the afterglow (column 3); and (iv) flares or bumps with a double break toward the end of the LC (column 4) and the models (top to bottom): (i) GP (row 1); (ii) Bi-Mamba model (row 2); (iii) MLP model (row 3); (iv) Fourier transform (row 4); and (v) GP+RF model (row 5).

when reducing computational complexity is a priority. Bi-Mamba also exhibits the lowest outliers for α of 1.15.

5. For $\log T_a$, Attention U-Net reduces the outliers by 1.88% compared to the MLP baseline. Similarly, for $\log F_a$, it achieves a 1.32% reduction, while for α, the reduction amounts to 0.49%.
6. The Fourier model, CGAN, and SARIMAX-based Kalman model, compared to the other models, demonstrate lower test MSE values (ranging from 0.0339 to 0.0825) but are not the best overall choices due to trade-offs in handling uncertainty, outliers, and model overfitting (as indicated by the discrepancy in the train and test losses).
7. The GP-RF and KAN models show higher test MSE and poor outlier control than the other tested models, making them less suitable for this specific reconstruction task.

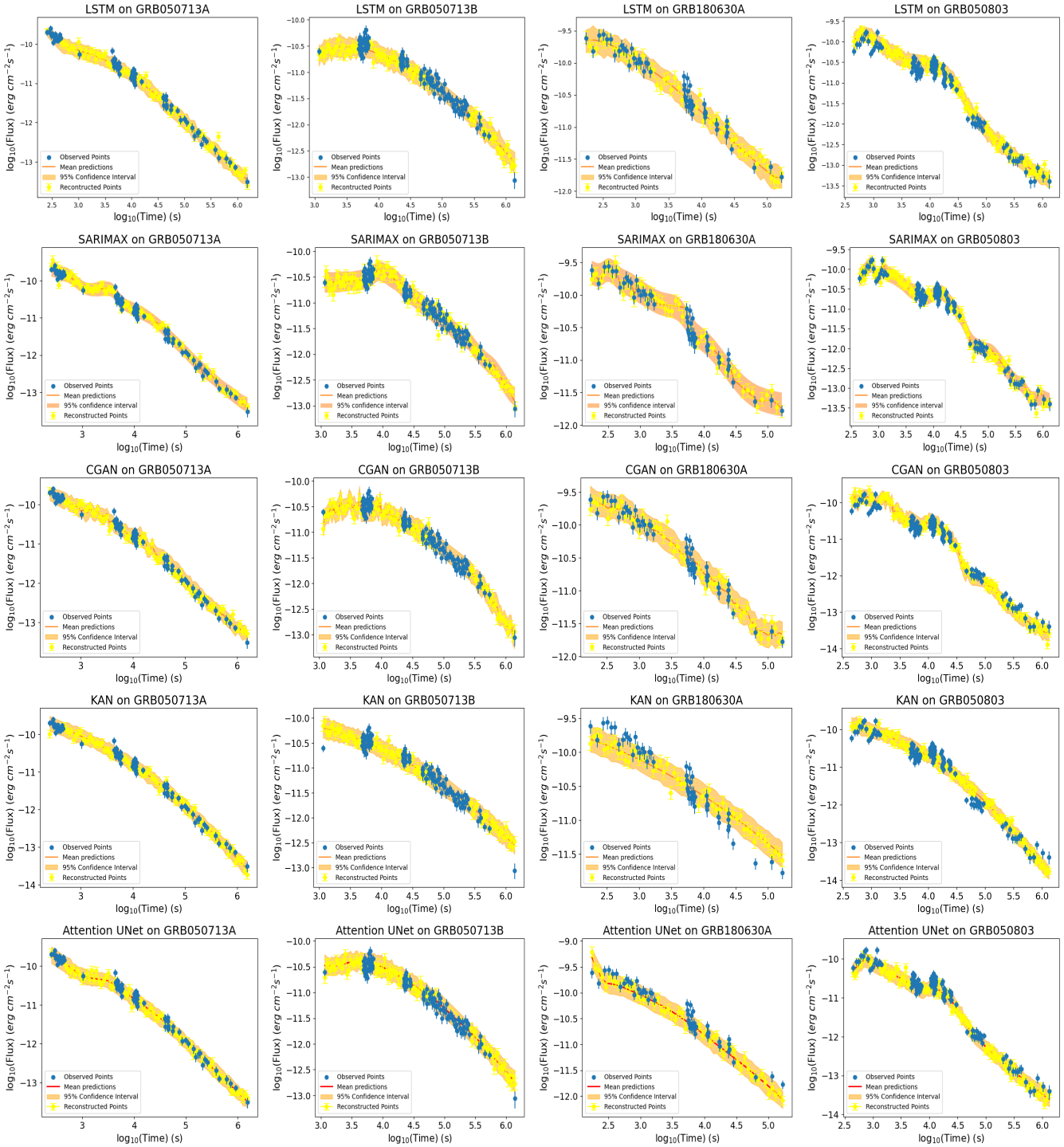


Figure 6. Reconstruction of LCs for all four varieties of GRBs are shown in a grid with four types of GRBs (left to right): (i) good GRBs (column 1); (ii) a GRB LC with a break toward the end (column 2); (iii) flares or bumps in the afterglow (column 3); (iv) flares or bumps with a double break toward the end of the LC (column 4) and the models (top to bottom): (i) Bi-LSTM model (row 1); (ii) SARIMA-based Kalman model (row 2); (iii) CGAN (row 3); (iv) KAN model (row 4); and (v) Attention U-Net model (row 5).

In summary, while Attention U-Net achieves the highest reduction in outliers and predictive uncertainty, its test MSE (0.134) is higher than that of the MLP. The MLP, in contrast, delivers the lowest test MSE and remains highly competitive in uncertainty reduction, with performance very close to Attention U-Net (within 0.5%–1.9% across all parameters). This balance between accuracy and stability suggests that the MLP provides a more reliable baseline for GRB LCR, while Attention U-Net offers advantages in minimizing extreme deviations.

In comparison to the prior study by M. G. Dainotti et al. (2023b), we have the following results:

1. As highlighted by M. G. Dainotti et al. (2023b), a study of 207 good GRBs revealed an average reduction in

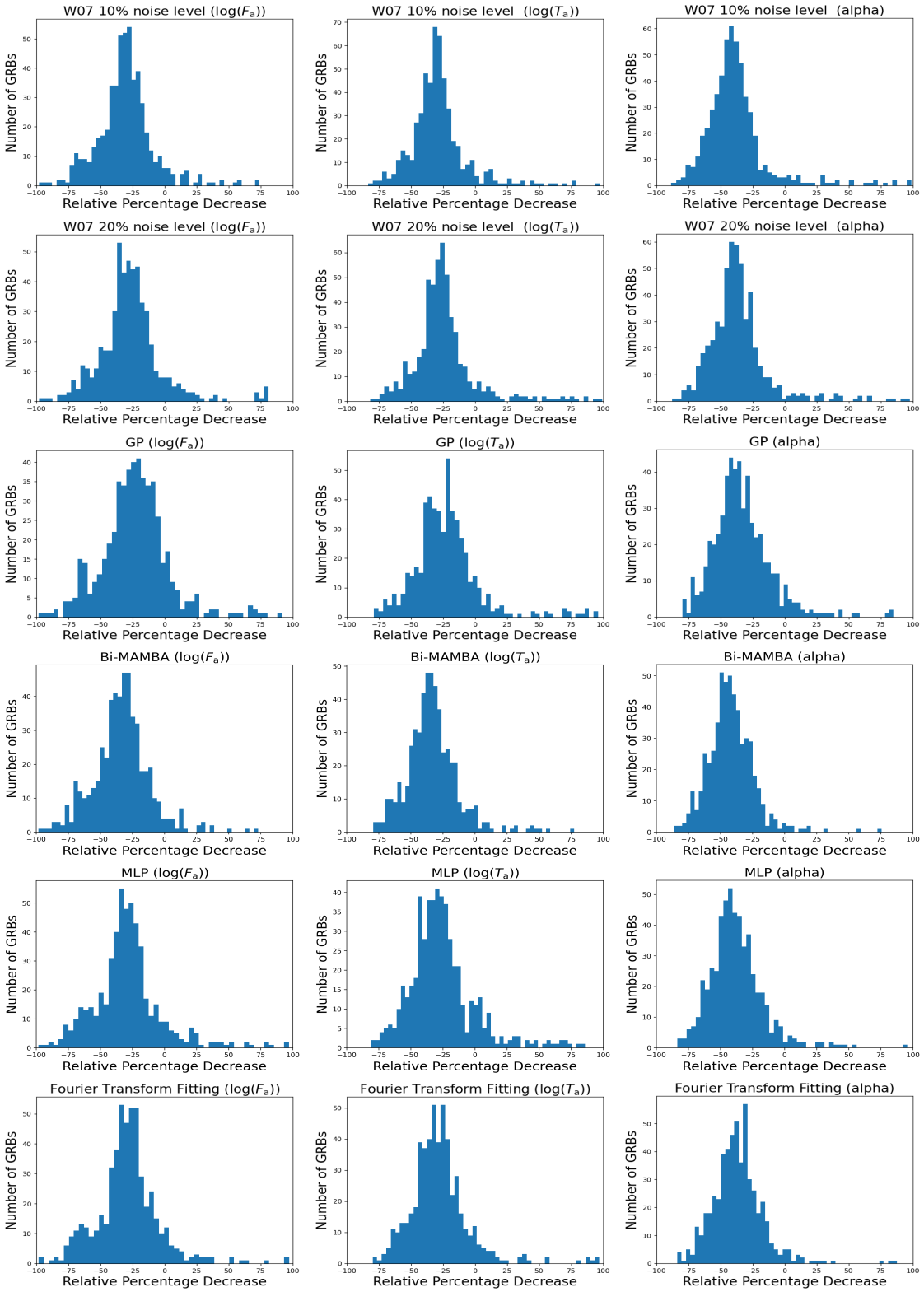


Figure 7. Distribution plot of all three W07 parameters in a grid with parameters (left to right): (i) log F_a (column 1); (ii) log T_a (column 2); (iii) α (column 3); and the models (top to bottom): (i) Willingale (10% noise) (row 1); (ii) W07 (20% noise) (row 2); (iii) GP model (row 3); (iv) Bi-Mamba model (row 4); (v) MLP model (row 5); and (vi) Fourier (row 6).

uncertainties of 26% across all parameters with a 10% noise level and 24% at a 20% noise level on modeling with the W07 model.

2. In our work, we include this exact data set of 207 GRBs for a fair comparison (see Table 5). When compared to the W07 model, which achieves uncertainty reductions of 23.0% for log T_a, 24.9% for log F_a, and 30.8% for α, Attention U-Net improves these values substantially to 38.8%, 40.3%, and 44.0%, respectively. This corresponds to relative improvements of 68.7%, 61.9%, and

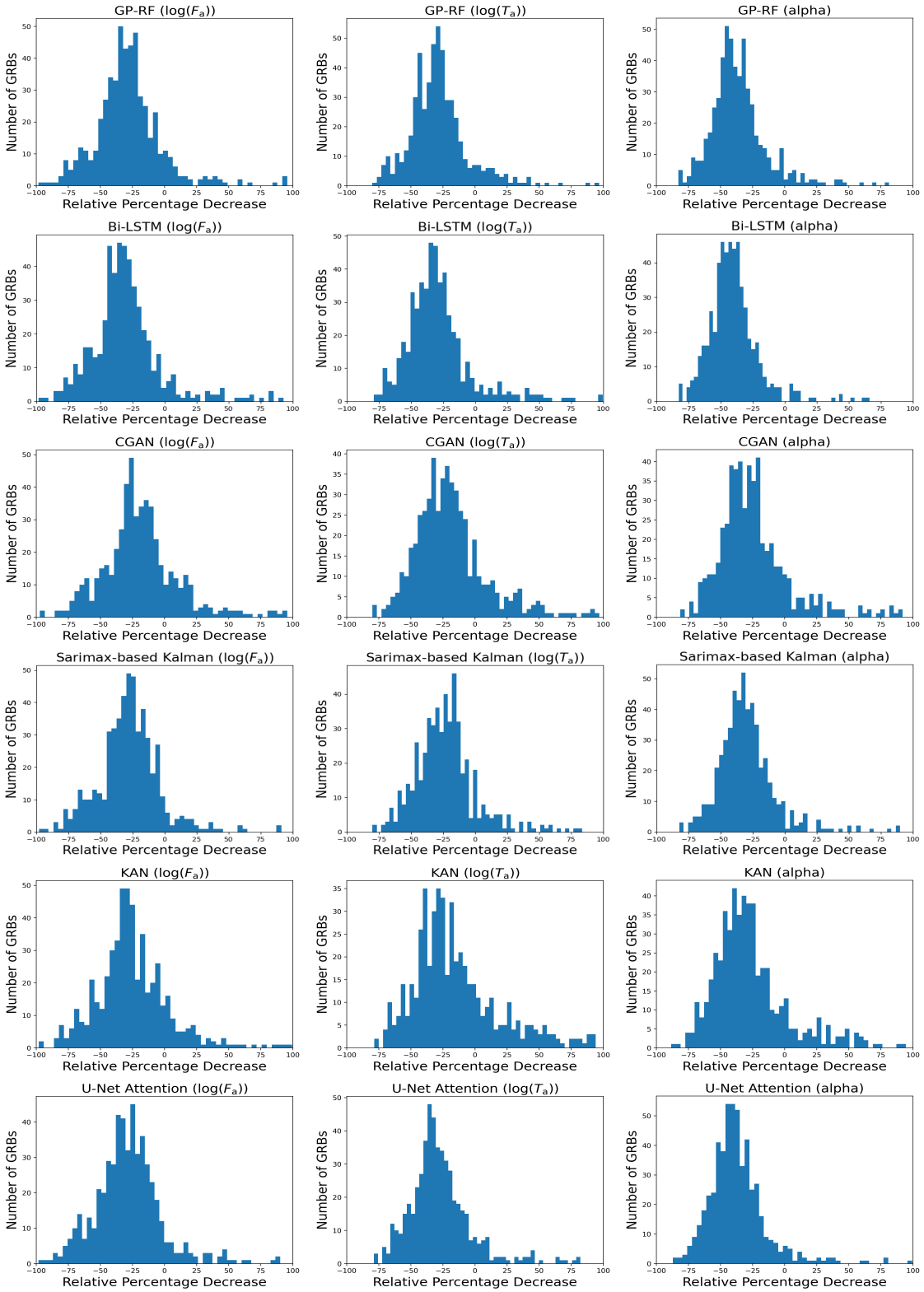


Figure 8. Distribution plot of all three W07 parameters in a grid with parameters (left to right): (i) log F_a (column 1); (ii) log T_a (column 2); (iii) α (column 3); and the models (top to bottom): (i) GP-RF (row 1); (ii) Bi-LSTM (row 2); (iii) CGAN (row 3); (iv) SARIMAX-based Kalman (row 4); (v) KAN (row 5); and (vi) Attention U-Net (row 6).

42.9%, indicating that Attention U-Net provides a more effective reduction of predictive uncertainty compared to the W07 model. Compared to the GP model, Attention U-Net outperforms for all three parameters. Additionally, the fivefold CV results show that both training and testing MSE values for Attention U-Net are lower than those of GP, further demonstrating its reliability. Another advantage of Attention U-Net is that it is model-independent while achieving performance equivalent to W07.

Table 5
Summary of Average Decrease in the Percentage Uncertainty along with the Percentage Increase in Outliers across Different Reconstruction Processes and Their Parameters

Reconstruction Model	Uncertainty Decrease			% Outliers			Five K-fold CV	
	% $\log T_a$	% $\log F_a$	% α	% $\log T_a$	% $\log F_a$	% α	Train MSE (10^{-1})	Test MSE (10^{-1}) ↓
				521 GRBs				
MLP	−37.2	−38.0	−41.2	**1.73**	2.30	1.34	0.227	**0.275**
Fourier	−14.9	−15.2	−20.6	4.61	4.22	2.50	**0.0270**	0.339
CGAN	−14.5	−15.4	−20.0	3.26	3.45	1.34	0.260	0.429
Bi-LSTM	−21.2	−21.5	−26.1	2.50	2.88	1.34	0.231	0.532
SARIMAX-based Kalman	−9.70	−13.3	−17.3	3.26	3.84	1.34	0.814	0.825
Bi-Mamba	−20.8	−21.2	−27.6	2.88	2.30	**1.15**	0.151	1.30
GP-RF	−16.3	−19.5	−23.2	4.03	4.03	2.30	0.0551	1.33
Attention U-Net	**−37.9**	**−38.5**	**−41.4**	**1.73**	2.50	1.34	0.206	1.34
KAN model	−4.74	−14.2	−11.6	4.99	**1.92**	1.73	0.423	1.74
GP (W07)	−16.9	−18.6	−24.3	3.07	3.45	1.54	8.56	3.63
W07 model (10%)	−18.0	−19.1	−25.2	2.30	2.30	2.11	⋯	⋯
W07 model (20%)	−15.8	−17.8	−23.8	2.30	2.69	2.30	⋯	⋯
				207 good GRBs (M. G. Dainotti et al. 2023b)				
SARIMAX-based Kalman	−10.9	−15.9	−19.8	0.966	1.45	**0**	0.191	**0.192**
MLP	−38.1	−39.3	−43.9	0.483	0.483	**0**	0.230	0.268
CGAN	−16.3	−17.9	−24.6	0.966	0.966	**0**	0.230	0.383
Fourier	−15.5	−17.3	−24.1	1.45	1.45	0.483	0.0900	0.480
Bi-LSTM	−22.7	−24.0	−30.1	0.483	0.483	**0**	0.210	0.626
Bi-Mamba	−22.5	−24.4	−31.2	0.483	0.483	**0**	0.110	0.850
GP-RF	−17.6	−21.8	−27.8	0.483	0.966	0.483	**0.0468**	0.920
Attention U-Net	**−38.8**	**−40.3**	**−44.0**	0.483	0.483	**0**	0.173	1.07
KAN model	−7.06	−17.3	−12.8	3.38	**0**	**0**	0.336	1.61
GP (W07)	−17.2	−20.0	−28.8	1.93	1.45	**0**	7.66	1.81
W07 model (10%)	−23.0	−24.9	−30.8	0.966	0.966	**0**	⋯	⋯
W07 model (20%)	−21.2	−23.2	−28.9	**0**	**0**	**0**	⋯	⋯

Note. The train MSE and test MSE for all the models across five K-fold CV are provided. The models are sorted in ascending order based on Test MSE values (top to bottom). GP, W07 (10%), and W07 (20%) are shown separately at the bottom for both 521 GRBs and 207 good GRBs. The bold values refer to the highest percentage decrease in the three parameters in the first three columns, the lowest percentage of outliers for the three parameters in columns (4), (5) and (6), and lowest test and train MSE for the last two columns.

3. This study effectively reduces parameter uncertainty for the subset of 207 well-behaved GRBs. However, when the approach is extended to the full data set encompassing all four GRB categories, the GP method exhibits reduced performance, as outlined in M. G. Dainotti et al. (2023b). In contrast, the other machine learning and DL models, particularly Attention U-Net, maintain a similar level of precision in reducing parameter uncertainty. This highlights the strength of our models in reconstructing more complex GRB LCs that include features such as flares or breaks. Furthermore, both training and testing MSE values remain low, demonstrating the continued reliability of the models. Although the number of outliers increases, from an average of 0.13% for the 207 GRBs to 2.5% across 521 GRBs, this is expected. This rise is expected as the Willingale function is not designed to model the behavior of more complex LCs.
4. In summary, this work undertakes a more comprehensive and challenging analysis than M. G. Dainotti et al. (2023b) by applying advanced machine learning and deep learning methods to a significantly larger and more diverse GRB data set. We provide more insight into the strengths and weaknesses of each approach by methodically evaluating a larger variety of GRBs, including those with intricate characteristics like flares and breaks. These advancements not only extend the scope of previous studies but also highlight the reliability and generalizability of our models.

The substantial decrease in uncertainties attained through these reconstructions enhances the applicability of the GRB plateau parameters for cosmological tools (M. G. Dainotti et al. 2022a, 2022b; M. Dainotti et al. 2023a).

Among the different evaluation aspects, the most important parameters in our analysis are $\log T_a$, $\log F_a$, and α, as they carry physical significance: α is crucial for theoretical modeling, as it describes the post-plateau decay behavior. This is relevant for the testing of the standard fireball model (T. Piran 1999), where the reduction of the α parameter is crucial to obtain reduced uncertainties on the closure relationship (G. P. Srinivasaragavan et al. 2020; M. G. Dainotti et al. 2021; M. Dainotti et al. 2025). It refers to the relationship between the spectral and temporal indices of a GRB's afterglow. $\log T_a$ and $\log F_a$ are key for identifying empirical correlations, such as the Dainotti relations in two and three dimensions (M. G. Dainotti et al. 2008, 2010, 2011, 2013, 2016, 2020b, 2020c, 2022c; D. Levine et al. 2022). The 2D Dainotti relation relates the end time of the plateau and its luminosity, while the three-dimensional relation adds

to the 2D relation the peak of the prompt emission parameter. These relations are of particular interest in cosmological applications and also for distinguishing among theoretical models.

Therefore, we consider these three parameters as the most important outputs of our reconstruction. Among all models tested, Attention U-Net demonstrates the most significant reduction in error for the three parameters. Thus, we recommend it as the preferred model when accurate estimation of these physically meaningful quantities is the main objective. The MSE is the second most important metric in our study. While it does not directly capture physical significance, it quantifies the overall performance and accuracy of the machine learning models, making it essential for model comparison. By this measure, the MLP performs best, delivering the lowest test MSE while still achieving uncertainty reductions that are very close to those of Attention U-Net. Outlier analysis is complementary and helps interpret cases where reconstruction might be less representative or behave anomalously.

In future research, we intend to explore advanced deep learning techniques, such as genetic algorithms, Bayesian neural networks, and neural ordinary differential equations. Moreover, we aim to investigate transformer models and large language models for their ability to identify complex temporal patterns. These approaches are expected to significantly improve LC reconstruction's accuracy, reliability, and applicability when combined with new data sets, particularly in theoretical model interpretation and cosmology.

This reconstruction framework has been developed exclusively using Swift LCs so far; however, we intend to extend its application to additional current missions such as SVOM (J.-L. Atteia et al. 2022) and the Einstein Probe (W. Yuan et al. 2022), as well as future missions, including THESEUS (L. Amati et al. 2018) and HiZ-GUNDAM (D. Yonetoku et al. 2024). Our extension will also incorporate data across multiple wavelengths. Expanding this work to optical wavelengths is especially timely, given the availability of the most extensive optical catalog to date (M. Dainotti et al. 2020a; M. Dainotti et al. 2025, 2024b; M. G. Dainotti et al. 2022c).

Acknowledgments

We want to thank Biagio De Simone for providing tips on the initial code for the reconstruction of the LC. The authors also thank Aditya Narendra, Nikita Khatiya, and Dhruv Bal for their insightful comments on the analysis of our models. We thank Spencer James Gibson and Federico Da Rold for their valuable insights on exploring different new algorithms for the reconstruction process. N.F. acknowledges financial support from UNAM-DGAPA-PAPIIT through the grant No. IN112525. M.G.D. acknowledges the support of the DoS and by JSPS Grant-in-Aid for Scientific Research (KAKENHI) (A), grant No. JP25H00675.

We thank Dr. Jurgen Mifsud, Dr. Purba Mukherjee, Dr. Konstantinos F. Dialektopoulos, and Prof. Jackson Said for their helpful comments and discussion of our analysis.

Appendix

This appendix explains the rationale for not using a separate test set to evaluate our model. It also highlights alternative models we experimented with, though they did not produce satisfactory results for our specific use case. However, we acknowledge that the model may hold more potential in different hybrid configurations for future work.

A.1. Test Set Evaluation

Due to the nature of our problem being data reconstruction rather than pure regression analysis, a separate test set cannot be effectively used. Each GRB must be fitted individually on the sparse time and flux values available to reconstruct the variability within its curve accurately. Attempts to train on multiple GRBs and reconstruct a separate test set led the model to reproduce a similar GRB structure across all test instances. This occurs because the model learns a single flux value for each corresponding timestamp and tends to generalize when reconstruction specific to each GRB is required. This behavior is evident in Figure 9, where Bi-Mamba, trained on GRB 050713A and tested on a separate set containing GRB 241026A and GRB 241127A, produces similar reconstructions for both test GRBs. This is because the model has access to sparse training data but only time samples from the test set.

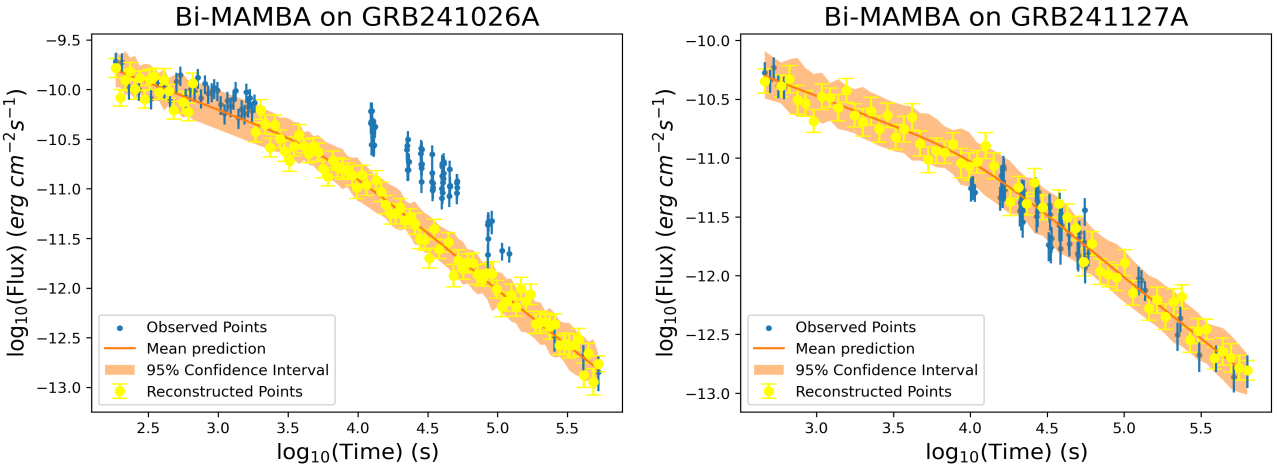


Figure 9. Reconstruction of LCs using Bi-Mamba trained on GRB 050713A. Left: reconstructed LC of GRB 241026A; right: reconstructed LC of GRB 241127A.

A.2. TimeAutoDiff

TimeAutoDiff (N. Suh et al. 2024) is a model designed to synthesize time-series tabular data, addressing the challenges of temporal dependencies and heterogeneous features. It combines a variational autoencoder (VAE; D. P. Kingma & M. Welling 2019) with a denoising diffusion probabilistic model (DDPM; J. Ho et al. 2020) to generate synthetic data.

The architecture of TimeAutoDiff consists of the following components:

1. *VAE component.* This leverages transformers to capture feature correlations and recurrent neural networks (RNNs) to model temporal dependencies in the data.
2. *DDPM component.* This module learns the data's latent distribution, enabling new samples with learned temporal structures.

While TimeAutoDiff demonstrated strong performance in reconstructing flux for timestamps it had already observed, it struggled to generalize to unseen timestamps, making it unsuitable for our reconstruction task.

A.3. VAEs

VAEs are generative models that are used for the reconstruction of data. Like other autoencoders, VAEs are made up of a decoder that reconstructs the input data from the latent variables and an encoder that learns to extract important latent variables from training data. The ability of VAEs to encode a continuous, probabilistic representation of the latent space rather than a fixed, discrete one distinguishes them from conventional autoencoders. VAEs can produce new data samples closely resembling the original data set through variational inference.

This probabilistic approach allows VAEs to explicitly capture and model the uncertainty inherent in the data. For GRB LCR, this capability is particularly advantageous, as it captures the variability and noise present in astrophysical observations while producing reliable reconstructions and confidence intervals for the predictions.

Despite their strengths, the VAEs struggled to accurately capture the temporal dependencies and the variability in the GRB LCs. However, we will continue exploring this method and believe that increasing the model's complexity, such as incorporating recurrent components or deeper architectures, could improve its ability to capture the intricate details of the LCs and enhance its performance.

A.4. Decision Trees

Decision trees (DTs) are a nonparametric supervised learning method mainly used for classification and regression. We use the classification and regression tree `CART` algorithm to build a tree, and the Python package scikit-learn for implementing DTs for regression.

DTs work by recursively splitting the data based on feature values to minimize the variance within the resulting subgroups. DTs are beneficial for LCR because they handle nonlinear relationships well. However, DTs tend to overfit the training data, especially when the LC has noisy measurements or complex temporal behavior. Although they perform well in capturing local variations, their predictions often lack the smoothness required for continuous processes like LCR. This limitation leads to step-like predictions rather than a smooth, gradual transition typically expected in astronomical LCs.

A.5. RFs

RFs are an ensemble method that builds upon the foundation of DTs by creating multiple trees using random subsets of the data and features and then averaging their predictions. This ensemble approach helps mitigate the overfitting issue commonly seen in single DTs, leading to improved performance.

RFs have one significant advantage over individual DTs: their ability to estimate uncertainty. Although RFs are not inherently probabilistic models, uncertainty can be calculated by measuring the variance of predictions across the ensemble of trees. RFs provide uncertainty quantification by examining the distribution of predictions from the individual trees.

However, while RFs offer some uncertainty estimation, they are still less precise than the uncertainty provided by fully probabilistic models like GPs. RF uncertainty is derived from the diversity among trees rather than the underlying data noise or model uncertainty, and the predicted intervals are less smooth and more step-like compared to probabilistic methods, as shown in Figure 10.

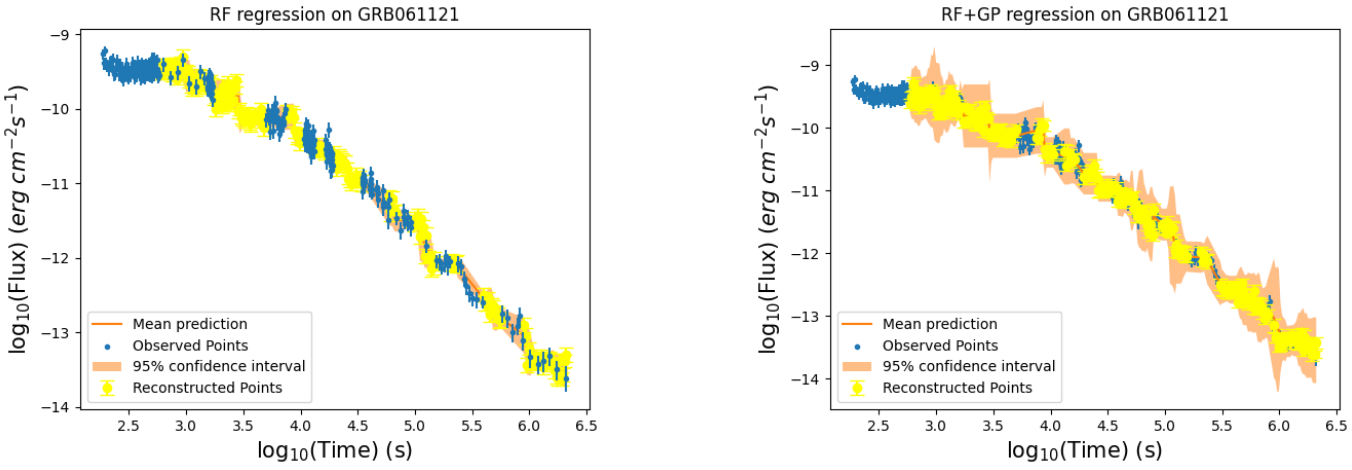


Figure 10. Reconstructed GRB LCs using (i) the RF model (left); (ii) incorporation of both the RF and GP regression models (right).

A.6. The Hybrid Model: RF with GP

To counter the step-like nature of RF predictions in LCR, we develop a hybrid model by combining RF with a GP. Here, the RF acts as the base model, trained on the data to capture initial predictions that efficiently handle nonlinear relationships. However, due to the RF's step-like nature, these predictions lack the smoothness typically required for reconstructing continuous time-series data.

Once the RF produces its predictions, they are passed to a GP for further modeling. The GP is trained on the RF predictions, learning a functional mapping that transforms the step-like predictions into smoother, continuous outputs. GPs are particularly well suited for this task because they can model uncertainty in a probabilistic framework while naturally producing smooth, continuous predictions. By integrating the GP, we aim to refine the overall prediction, achieving a smoother representation of the LC with a continuity that better reflects its underlying structure.

However, a limitation of this approach is that the error propagation from the RF propagates via the GP, and the step-like uncertainty patterns remain prominent in the hybrid model's confidence intervals. While the mean predictions become smoother, the uncertainty estimates retain a step-like nature, as shown in Figure 10. Thus, this hybrid model improves the smoothness of the LC reconstruction but still lacks the fully continuous uncertainty representation characteristic of models with an entirely probabilistic approach. In Section 2.3, we introduce another hybrid approach that reverses the base structure: instead of using RF as the foundation, we employ GP as the base model and combine it with RF. In this, GP is first trained on the observed data to generate smooth, uncertainty-aware predictions. The RF acts as a secondary learner that attempts to correct residual patterns or finer-scale structures that the GP might have smoothed out or underfit. Since the RF now operates on a smoother input (from GP), the resulting uncertainty propagation becomes more consistent and less step-like.

ORCID iDs

A. Manchanda https://orcid.org/0009-0000-9379-6279
M. G. Dainotti https://orcid.org/0000-0003-4442-8546
D. H. Hartmann https://orcid.org/0000-0002-8028-0991
A. Pollo https://orcid.org/0000-0003-3358-0665
J. X. Prochaska https://orcid.org/0000-0002-7738-6875
N. Fraija https://orcid.org/0000-0002-0173-6453

References

Agarap, A. 2018, arXiv:1803.08375
Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. 2019, in Proc. 25th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (New York: ACM), 2623
Alharbi, F. R., & Csala, D. 2022, Inventions, 7, 94
Amati, L., O'Brien, P., Götz, D., et al. 2018, AdSpR, 62, 191
Aravkin, A., Burke, J., & Pillonetto, G. 2013, SJCO, 52, 2891
Atteia, J.-L., Cordier, B., & Wei, J. 2022, IJMPD, 31, 2230008
Bargiacchi, G., Dainotti, M. G., & Capozziello, S. 2025, NewAR, 100, 101712
Barthelmy, S. D., Barbier, L. M., Cummings, J. R., et al. 2005, SSRv, 120, 143
Beskin, G., Karpov, S., Bondar, S., et al. 2010, ApJL, 719, L10
Betoule, M., Kessler, R., Guy, J., et al. 2014, A&A, 568, A22
Blake, C. H., Bloom, J. S., Starr, D. L., et al. 2005, Natur, 435, 181
Breiman, L. 2001, MachL, 45, 5
Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. 2017, Classification and Regression Trees (London: Chapman and Hall/CRC)
Brophy, E., Wang, Z., She, Q., & Ward, T. 2023, ACM Comput. Surv., 55, 1
Burrows, D. N., Hill, J. E., Nousek, J. A., et al. 2005, SSRv, 120, 165
Cao, S., Dainotti, M., & Ratra, B. 2022a, MNRAS, 512, 439
Cao, S., Khadka, N., & Ratra, B. 2022b, MNRAS, 510, 2928
Cardone, V. F., Capozziello, S., & Dainotti, M. G. 2009, MNRAS, 400, 775
Cardone, V. F., Dainotti, M. G., Capozziello, S., & Willingale, R. 2010, MNRAS, 408, 1181
Cheng, J., Dong, L., & Lapata, M. 2016, arXiv:1601.06733
Costa, E., Frontera, F., Heise, J., et al. 1997, Natur, 387, 783
Cucchiara, A., Levan, A. J., Fox, D. B., et al. 2011, ApJ, 736, 7
Dainotti, M., Bhardwaj, S., Bissaldi, E., et al. 2025, ApJ, 978, 51
Dainotti, M., Lenart, A., Chraya, A., et al. 2023a, MNRAS, 518, 2201
Dainotti, M., Livermore, S., Kann, D., et al. 2020a, ApJL, 905, L26
Dainotti, M., Taira, E., Wang, E., et al. 2024a, ApJS, 271, 22
Dainotti, M., De Simone, B., Mohideen Malik, R., et al. 2024b, MNRAS, 533, 4023
Dainotti, M. G., Cardone, V. F., & Capozziello, S. 2008, MNRAS, 391, L79
Dainotti, M. G., De Simone, B., Schiavone, T., et al. 2022a, Galax, 10, 24
Dainotti, M. G., Del Vecchio, R., Shigehiro, N., & Capozziello, S. 2015, ApJ, 800, 31
Dainotti, M. G., Fabrizio Cardone, V., Capozziello, S., Ostrowski, M., & Willingale, R. 2011, ApJ, 730, 135
Dainotti, M. G., Lenart, A., Fraija, N., et al. 2021, PASJ, 73, 970
Dainotti, M. G., Lenart, A., Sarracino, G., et al. 2020b, ApJ, 904, 97
Dainotti, M. G., Livermore, S., Kann, D. A., et al. 2020c, ApJL, 905, L26
Dainotti, M. G., Nagataki, S., Maeda, K., Postnikov, S., & Pian, E. 2017, A&A, 600, A98
Dainotti, M. G., Nielson, V., Sarracino, G., et al. 2022b, MNRAS, 514, 1828
Dainotti, M. G., Postnikov, S., Hernandez, X., & Ostrowski, M. 2016, ApJL, 825, L20
Dainotti, M. G., Sarracino, G., & Capozziello, S. 2022c, PASJ, 74, 1095
Dainotti, M. G., Sharma, R., Narendra, A., et al. 2023b, ApJS, 267, 42
Dainotti, M. G., Singal, J., Ostrowski, M., et al. 2013, ApJ, 774, 157
Dainotti, M. G., Willingale, R., Capozziello, S., Fabrizio Cardone, V., & Ostrowski, M. 2010, ApJL, 722, L215
Dainotti, M. G., Young, S., Li, L., et al. 2022d, ApJS, 261, 25
Demianenko, M., Malanchev, K., Samorodova, E., et al. 2023, A&A, 677, A16
Dereli-Bégué, H., Pe'er, A., Bégué, D., & Ryde, F. 2025, ApJ, 985, 261
Evans, P., Beardmore, A., Page, K., et al. 2009, MNRAS, 397, 1177
Fu, R., Chen, J., Zeng, S., Zhuang, Y., & Sudjianto, A. 2019, arXiv:1904.11419
Gehrels, N., Chincarini, G., Giommi, P., et al. 2004, ApJ, 611, 1005
Gehrels, N., Ramirez-Ruiz, E., & Fox, D. B. 2009, ARA&A, 47, 567
Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., et al. 2014, in Advances in Neural Information Processing Systems 27, ed. Z. Ghahramani (NeurIPS), 2672, https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf
Gorbovskoy, E. S., Lipunova, G. V., Lipunov, V. M., et al. 2012, MNRAS, 421, 1874
Graves, A., Fernández, S., & Schmidhuber, J. 2005, in Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005, ed. W. Duch et al. (Berlin: Springer), 799
Gu, A., & Dao, T. 2023, arXiv:2312.00752
He, K., Zhang, X., Ren, S., & Sun, J. 2015, in IEEE Int. Conf. on Computer Vision (ICCV) (Piscataway, NJ: IEEE), 1026
Heaton, J. 2018, Genet. Progr. Evolvable Mach., 19, 305
Ho, J., Jain, A., & Abbeel, P. 2020, in Advances in Neural Information Processing Systems 33 (NeurIPS), 6840, https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf
Ho, S., & Xie, M. 1998, Comput. Ind. Eng., 35, 213
Hochreiter, S., & Schmidhuber, J. 1997, Neural Comput., 9, 1735
Hua, Y., Zhao, Z., Li, R., et al. 2019, IComM, 57, 114
Huber, S., & Suyu, S. 2024, A&A, 692, A132
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Kingma, D. P., & Welling, M. 2019, Found. Trends Mach. Learn., 12, 307
Kohavi, R. 1995, in Proc. 14th Int. Joint Conf. on Artificial Intelligence (IJCAI) (San Francisco, CA: Morgan Kaufmann), 1137
Kumar, P., & Duran, R. B. 2010, MNRAS, 409, 226
Kumar, P., & Zhang, B. 2015, PhR, 561, 1
Levine, D., Dainotti, M., Zvonarek, K. J., et al. 2022, ApJ, 925, 15
Li, L., Wu, X.-F., Lei, W.-H., et al. 2018, ApJS, 236, 26
Liang, E.-W., Zhang, B.-B., & Zhang, B. 2007, ApJ, 670, 565
Liu, H., Liu, C., Wang, J. T., & Wang, H. 2019, ApJ, 877, 121
Liu, Z., Wang, Y., Vaidya, S., et al. 2024, arXiv:2404.19756

Medsker, L. R., Jain, L., et al. 2001, Recurrent Neural Networks: Design and Applications, Vol. 5 (1st ed.; Boca Raton, FL: CRC Press), 2
Mirza, M., & Osindero, S. 2014, arXiv:1411.1784
Narendra, A., Dainotti, M., Sarkar, M., et al. 2025, A&A, 698, A92
Nousek, J. A., Kouveliotou, C., Grupe, D., et al. 2006, ApJ, 642, 389
Oberst, U. 2007, SJCO, 46, 496
O'Brien, P. T., Willingale, R., Osborne, J., et al. 2006, ApJ, 647, 1213
Oktay, O., Schlemper, J., Folgoc, L. L., et al. 2018, arXiv:1804.03999
Panaitescu, A., & Kumar, P. 2000, ApJ, 543, 66
Pechlivanidou, G., & Karampetakis, N. 2022, IMA J. Math. Control Inform., 39, 708
Perslev, M., Jensen, M., Darkner, S., Jennum, P. J., & Igel, C. 2019, in Advances in Neural Information Processing Systems 32, ed. H. Wallach et al. (NeurIPS) https://proceedings.neurips.cc/paper_files/paper/2019/file/57bafb2c2dfeefba931bb03a835b1fa9-Paper.pdf
Piran, T. 1999, PhR, 314, 575
Piro, L., Amati, L., Antonelli, L. A., et al. 1998, A&A, 331, L41
Postnikov, S., Dainotti, M. G., Hernandez, X., & Capozziello, S. 2014, ApJ, 783, 126
Racusin, J. L., Liang, E. W., Burrows, D. N., et al. 2009, ApJ, 698, 43
Rasmussen, C. E. 2003, in Advanced Lectures on Machine Learning. ML 2003, ed. O. Bousquet, U. von Luxburg, & G. Rätsch (Berlin: Springer), 63
Rea, N., Gullón, M., Pons, J. A., et al. 2015, ApJ, 813, 92
Roming, P. W. A., Kennedy, T. E., Mason, K. O., et al. 2005, SSRv, 120, 95
Ronneberger, O., Fischer, P., & Brox, T. 2015, in Medical Image Computing and Computer-assisted Intervention–MICCAI 2015, ed. N. Navab et al. (Cham: Springer), 234
Rowlinson, A., Gompertz, B. P., Dainotti, M., et al. 2014, MNRAS, 443, 1779
Ryan, G., Van Eerten, H., Piro, L., & Troja, E. 2020, ApJ, 896, 166
Sakamoto, T., Hill, J. E., Yamazaki, R., et al. 2007, ApJ, 669, 1115
Srinivasaragavan, G. P., Dainotti, M. G., Fraija, N., et al. 2020, ApJ, 903, 18
Stratta, G., Dainotti, M. G., Dall'Osso, S., Hernandez, X., & De Cesare, G. 2018, ApJ, 869, 155
Suh, N., Yang, Y., Hsieh, D.-Y., et al. 2024, arXiv:2406.16028
Tagliaferri, G., Goad, M., Chincarini, G., et al. 2005, Natur, 436, 985
Tak, D., Omodei, N., Uhm, Z. L., et al. 2019, ApJ, 883, 134
Tang, C.-H., Huang, Y.-F., Geng, J.-J., & Zhang, Z.-B. 2019, ApJS, 245, 1
Thielmann, A. F., Kumar, M., Weisser, C., et al. 2024, arXiv:2408.06291
Troja, E., Cusumano, G., O'Brien, P. T., et al. 2007, ApJ, 665, 599
van Paradijs, J., Groot, P. J., Galama, T., et al. 1997, Natur, 386, 686
Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, arXiv:1706.03762
Vestrand, W. T., Wozniak, P. R., Wren, J. A., et al. 2005, Natur, 435, 178
Vestrand, W. T., Wren, J. A., Panaitescu, A., et al. 2014, Sci, 343, 38
Wang, F. Y., Hu, J. P., Zhang, G. Q., & Dai, Z. G. 2022, ApJ, 924, 97
Wang, S.-C. 2003, Interdisciplinary Computing in Java Programming (Boston, MA: Springer), 81
Willingale, R., O'brien, P., Osborne, J., et al. 2007, ApJ, 662, 1093
Xu, B., Wang, N., Chen, T., & Li, M. 2015, arXiv:1505.00853
Yonetoku, D., Doi, A., Mihara, T., et al. 2024, Proc. SPIE, 13093, 618
Yuan, W., Zhang, C., Chen, Y., & Ling, Z. 2022, in Handbook of X-ray and Gamma-ray Astrophysics, ed. C. Bambi & A. Santangelo (Singapore: Springer), 86
Zhang, B., Fan, Y. Z., Dyks, J., et al. 2006, ApJ, 642, 354
Zhang, B., & Mészáros, P. 2001, ApJL, 552, L35
Zhang, B., Zhang, B.-B., Liang, E.-W., et al. 2007, ApJL, 655, L25
Zhang, Y., Chen, G., Yu, D., et al. 2016, in Int. Conf. on Acoustics, Speech and Signal Processing (Piscataway, NJ: IEEE), 5755
Zhao, L., Zhang, B., Gao, H., et al. 2019, ApJ, 883, 97